NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

CLASSIFICATION OF DIGITAL MODULATION TYPES IN MULTIPATH ENVIRONMENTS

by

George Hatzichristos

March 2001

Thesis Advisor: Monique P. Fargues

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 2001	3. REPORT TYPE AND DATES COVERED Electrical Engineer's Thesis	
4. TITLE AND SUBTITLE **Classification of Digital Modulation Types in Multipath Environments**			5. FUNDING NUMBERS
6. AUTHOR(S) Hatzichristos, George			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

As the expansion of digital communication applications still continues, the need for automated classification of digital modulation types increases. This study attempts to give a partial solution to this problem by proposing a classification scheme which identifies nine of the most popular digital modulation types; namely 2-FSK, 4-FSK, 8-FSK, 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM and 256-QAM. Higher-order statistics parameters are selected as class features, and a hierarchical neural network-based classifier set-up proposed for the identification of all modulation types considered except those within the M-QAM family. Specific M-QAM types identification is obtained via equalization-based schemes. This study considers the effects due to real-world multipath propagation channels and additive white Gaussian noise. Results show a consistent overall classification performance of at least 68% for severe multipath propagation models and for SNR levels as low as 11dB.

14. SUBJECT TERMS Digital modulations,Propagation channels, Moments, Cumulants, Neural Networks, Classifier			15. NUMBER OF PAGES 226
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

CLASSIFICATION OF DIGITAL MODULATION TYPES IN MULTIPATH ENVIRONMENTS

George Hatzichristos
Lieutenant Junior Grade, Hellenic Navy
B.S, Hellenic Navy Academy, 1993

Submitted in partial fulfillment of the
requirements for the degree of

ELECTRICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
March 2001

Author: [illegible]
George Hatzichristos

Approved by: [illegible]
Monique P. Fargues, Thesis Advisor

[illegible]
Tri Ha, Committee Member

[illegible]
Roberto Cristi, Committee Member

[illegible]
Jeffrey B. Knorr, Chairman
Department of Electrical and Computer Engineering

THIS PAGE   INTENTIONALLY LEFT BLANK

ABSTRACT

As the expansion of digital communication applications still continues, the need for automated classification of digital modulation types increases. This study attempts to give a partial solution to this problem by proposing a classification scheme which identifies nine of the most popular digital modulation types; namely 2-FSK, 4-FSK, 8-FSK, 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM and 256-QAM. Higher-order statistics parameters are selected as class features, and a hierarchical neural network-based classifier set-up proposed for the identification of all modulation types considered except those within the M-QAM family. Specific M-QAM types identification is obtained via equalization-based schemes. This study considers the effects due to real-world multipath propagation channels and additive white Gaussian noise. Results show a consistent overall classification performance of at least 68% for severe multipath propagation models and for SNR levels as low as 11dB.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE   INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

EXECUTIVE SUMMARY

Classifying signal types is of high interest in numerous different application areas, as classification problems arise in commercial and military areas such as imaging, communication, control, target recognition, etc.... This thesis specifically focuses on the classification of various commonly used digital modulation types, namely 2-PSK, 4-PSK, 8-PSK, 2-FSK, 4-FSK, 8-FSK, 16-QAM, 64-QAM and 256-QAM.

Numerous research results have already been presented in this area, however most of them deal with either a small number of symbol states, relatively clean channel characteristics, or require large amounts of data. A significant aspect of this thesis was the design of a robust classification scheme that is able to work even under unfavorable propagation conditions such as realistic multipath and fading, and noise distortions. A hierarchical classification approach was selected as it allows for a relatively simple overall scheme with only few parameters needed to differentiate between modulation types. Back propagation neural network units were adopted because they offer the flexibility needed to cope with propagation environments that constantly change, as is the case in real-world communications.

The selection of higher-order statistics parameters as class features for the neural network classification units proved to be effective and robust for all classification schemes, except for specific M-QAM modulation types. Simulations showed that M-QAM types may be so affected by multipath and fading that higher-order statistic parameters become of very limited use for their specific identification. Equalization algorithms proved to be the only solution to separate M-QAM signals with satisfying results in medium to high SNR levels. The equalization step combines the generic blind equalization CMA-FSE algorithm and the constellation-specific Alphabet Matched equalization algorithm, and differentiates between high-order QAM modulation types in medium to high SNR environments.

The overall classifier was intensively tested in various propagation situations and signal-to-noise ratio (SNR) levels. Simulations show classification performances may be strongly affected by the amount of multipath distortion and noise in the transmission channels. Results show overall classification performances of 99% at 20dB down to 90% at 8dB for rural area propagation environments, while in more highly distorted channels such as urban propagation environments, overall classification performances are 82% at 20dB down to 68% at 11dB.

Results also showed that the greatest difficulty of this study was in the separation of M-QAM modulation types, which proved to be much more sensitive to channel degradations than the other modulation types considered.

ACKNOWLEDGEMENTS

This study is dedicated to my grandmother, Anna. She has always inspired me to accomplish whatever I have done in my life until now. Thank you grandma…
I would also like to express my gratitude to Elena, my parents and sister who have always been by my side through good and bad times.
Finally, I would like to express my deep respect for my thesis advisor, Professor Monique Fargues. Without her help and insistence this work would not have been done.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. THESIS OBJECTIVES

Digital telecommunications have been introduced in our everyday lives, from cellular telephony to satellite communications and from fast wireless Internet access to remote missile guidance systems. But as this evolution continues, unexpected problems, such as congested bandwidth, have come up. Classifying signal types is of high interest in numerous different application areas. Classification problems arise in commercial and military areas such as imaging, communication, control, target recognition, etc.... This thesis specifically focuses on the classification of digital modulation types M-PSK, M-FSK, M-QAM. A significant body of work exists in this area, however most of it deals with either a small number of symbol states M, relatively clean channel characteristics, or requires large amounts of data. This work first investigates the selection of robust and well-defined class features, and next designs a classification procedure which can be applied under some extreme conditions such as low SNR levels, realistic fading and real-world multi-path propagation channels.

B. THESIS ORGANIZATION

Following an overview of the most recent research results available in the open literature in the area of digital modulation classification, Chapter II introduces the basic concept behind a digital communication system. Next, we briefly review the commonly used modulation schemes considered in this study, namely M-FSK, M-PSK and M-QAM. Chapter III discusses propagation issues and explains how they may affect transmitted signal quality. In Chapter IV, the concept of signal equalization is defined, and two iterative methods used in this study are analytically described and tested. Chapter V introduces the concept of higher-order statistics and specifically focuses on higher-order statistical moments and cumulants as they are selected as identification features in the classification set-up. We investigate the robustness of these features with respect to propagation problems. Chapter VI describes the basic principles of neural networks, as they form the core of the proposed classification scheme. In Chapter VII the proposed classifier is analytically described and evaluated with extensive simulations. Chapter VIII summarizes the results and presents recommendations for further extensions.

C. BACKGROUND

The recognition of digital modulation types has been investigated extensively over the last twenty years. Numerous different approaches using the time and/or the frequency domain have been proposed, and those can be subdivided in two main general families; decision-theoretical or statistical pattern recognition approaches. This section briefly reviews some of the most recent work done in these two areas.

Soliman and Hsue proposed to use statistical moments as class features to classify CW and M-PSK signals [SAH92]. Their approach achieves classification performances close to 100% for SNR levels greater than 10dB. However, no simulation on real world propagation models has been reported.

Azzouz and Nandi proposed a hierarchical tree-based approach to classify constant amplitude signals such as CW, M-PSK and M-FSK [ANA95]. The selected features are statistical characteristics such as the power density or the standard deviation of the normalized centered instantaneous amplitude of the signal, etc.... Results showed this simple scheme to have 90% correct classification rates for signals in additive white Gaussian noise at SNR levels equal to 10dB or higher. However, once again, no simulation results for signals transmitted using real-world "types" of propagation channels situations have been reported with their approach.

Polydoros, Long and Chugg present a maximum likelihood approach to the problem [PLC96]. Their method classifies OQPSK, BPSK and QPSK modulation types contaminated with additive Gaussian white noise, using decision rules based on a maximum likelihood criteria. Results show a 99% correct classification rate for SNR levels equal to 5dB or higher. However, their scheme requires some a-priori signal information, such as the initial phase and symbol rate, making the whole process less practical.

Ketterer, Jondral and Costa propose a time-frequency approach to the problem [KJC99]. This scheme is a two-step process. First, autoregressive modeling is used to estimate the carrier frequency. Next, the time-frequency information, provided by the Cross-Margenau-Hill distribution [HPC95], is applied to estimate phase shifts, frequency shifts and amplitude shifts allowing the separation of M-PSK, M-FSK and M-QAM signals respectively. Simulations show modulation classification performance over 97% for SNR levels larger than 10dB. Unfortunately no simulations under real world propagation channels were made to further test the robustness of the method.

Huo and Donoho propose a different method to classify 4-QAM and 6-PSK [HAD98]. They use a classifier designed to minimize the Hellinger distance [BER77] between the empirical distribution of the intercepted signal and the true signal density function. The proposed scheme leads to recognition performances equal to 100% for SNR levels equal to 15dB or higher. However, such performance requires the knowledge of the channel model, and recognition performances drop significantly when dealing with unknown channels.

Hong and Ho apply wavelet transforms to classify M-PSK, M-FSK and M-QAM signals [HAH99]. Their simulation focuses on 16-QAM, Q-PSK and Q-FSK and gives correct classification percentage of over 98% for SNR levels equal to 5dB and higher. However, no simulations under fading multi-path propagation conditions are available, once again.

Mobasseri considers a pattern recognition approach and uses the constellation shape information obtained from the received signal to estimate the digital modulation type by applying fuzzy c-means cluster analysis [MOB00]. This scheme works well to separate low order constellations such as QPSK, 8-PSK and 16-QAM and provides correct recognition of over 90% for signal-to-noise ratios larger than 5dB.,However, no results are provided for signals transmitted over real-world propagation channels that might rotate and severely distort the signal's constellation.

Marchand, Martret and Lacoume use cumulants and moments to build a matched filter classification system that has an exceptional performance, close to 100% of accurate recognition for SNR levels equal to 0dB or higher [MML97]. This classifier is tested to identify 4-PSK versus 16-QAM but may easily be modified to incorporate more modulation schemes. However, again no simulation on fading multipath channels is conducted. This work is extended later by Marchand who selects moments and cumulants for classifying purposes in his Dissertation work [MAR98]. He proposes a computational inexpensive scheme to classify M-PSK and M-QAM signal types, and investigates the robustness of the scheme with respect to varying level of additive noise and number of symbols.

Ananthram Swami and Brian M. Sadler [SAS00] also select cumulants for class features. They introduce a hierarchical tree classifier scheme that uses second and fourth order cumulants to classify M-PSK, PAM, and M-QAM signals and achieves a 100% correct recognition for SNR levels higher than 8dB, where the number of states M is 4, 8, 16, and 32. Their encouraging conclusion is that the method may easily be expanded to a higher level of constellations such as 64-QAM, by increasing the order of the cumulants selected for class features.

Finally, Barbarossa, Swami, Sadler and Spadafora recently proposed the Alphabet Matched Algorithm (AMA), which is an iterative gradient descent scheme where the cost function to be minimized is based on a pre-determined signal constellation structure M-PSK and M-QAM signals [BAR00]. Their results show that the AMA is able to classify higher order constellations (such as 64-QAM) propagated through a linear channel in

SNR levels of 30dB perfectly. This method is further analyzed and implemented in this study [Chapter IV, Sections B, C].

D. REQUIRED SOFTWARE

MATLAB, version 5.3 was used to generate the data and conduct the simulations while EXCEL 2000 has been utilized to store all simulation results. We attempted to duplicate real-world conditions by selecting transmission channel models obtained from field measurements [RIC00]. Further details regarding the transmission channel types considered are presented in Appendix C.

II. DIGITAL COMMUNICATION SYSTEMS AND MODULATION SCHEMES

This chapter presents a brief overview of basic communication systems and popular digital modulation schemes.

A. INTRODUCTION TO DIGITAL COMMUNICATION SYSTEMS

With the explosion in the computer industry of the last fifteen years we now have the ability to process digital information with speeds that no one could have ever imagined a few decades ago. The basics of a digital communication system are described in Figure II-1. Communication basically means transmission of binary information sequences $\{b_k\}$. Such sequences are encoded prior to transmission to make the transmitted signal more robust to noise, interference and other channel degradations. Next, the resulting signal $d_k(t)$ is modulated by a sinusoidal carrier and passed through a transmitter filter to limit the signal bandwidth prior to transmission. The transmitted signal $s_k(t)$ does not normally reach the receiver without distortion, which can be due to white gaussian, colored noise or other narrowband signal interferences.

This study will utilize baseband signals exclusively, as heterodyning down transmitted signals is usually conducted at the receiver to decrease the needed sampling rates prior to further processing. Therefore, we will assume that the carrier has been estimated correctly and that no distortion in the received signal is produced as a result of estimation errors in the carrier frequency.

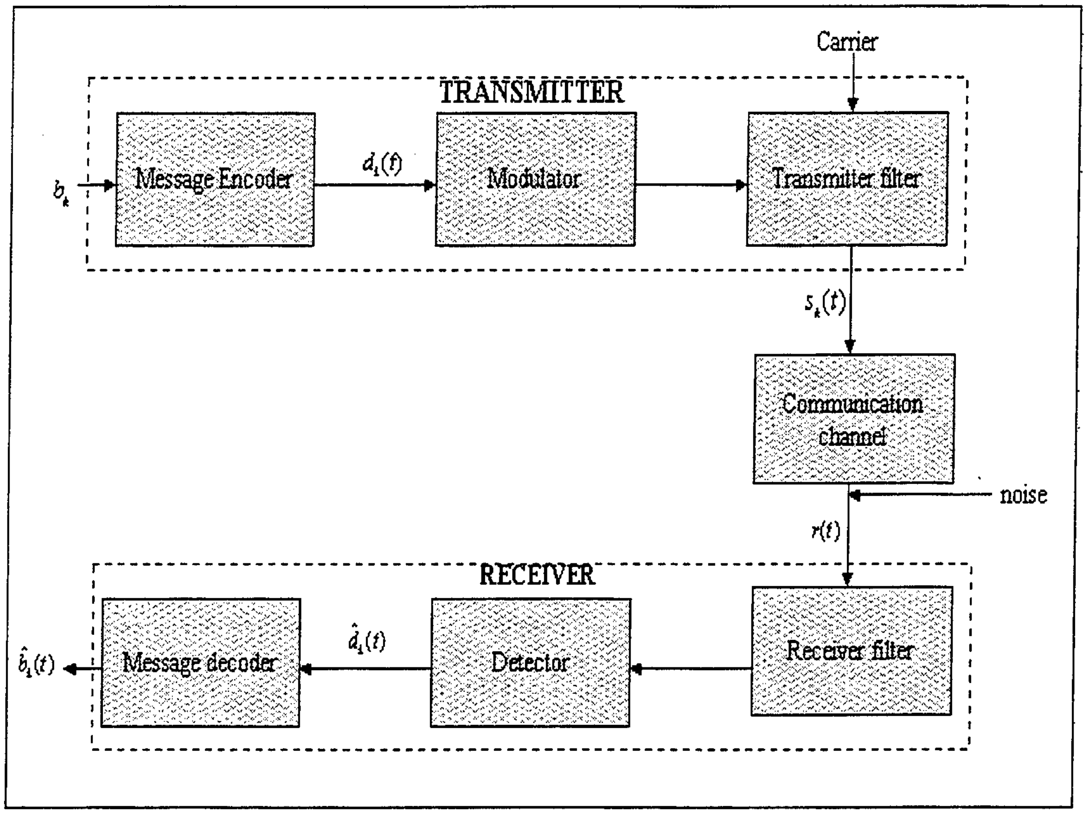


Figure II-1. Digital Communication System Model.

B. DIGITAL MODULATION TECHNIQUES

1. Introduction

Almost all modern communication systems use digital modulation techniques as they have many advantages over analog modulation schemes. For instance, digital modulation techniques offer greater noise immunity and robustness to channel distortions, easier multiplexing of various forms of information (e.g. voice and data), and greater security [MPRG00]. Several factors influence the choice of a digital modulation scheme. Ideally, a desirable modulation scheme provides low bit error rates at low received signal-to-noise ratios, has a good performance in multipath and fading conditions, occupies a minimum bandwidth, and is easy and cost-effective to implement. Existing modulation schemes do not simultaneously satisfy all of these requirements. Some are better in terms of the bit error rate performance, while others are better in terms of bandwidth efficiency. As a result, trade-offs need to be made when selecting a digital modulation depending on the demands of the particular application. For example, higher level modulation schemes (M-ary keying) require small bandwidth but higher received power than other schemes.

Some of the most widely used digital modulation techniques are summarized in Table II-1 below. This study will concentrate on 2-FSK, 4-FSK, 8-FSK, 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM and 256-QAM modulation schemes.

Linear Modulation Techniques	**Constant Envelope Modulation Techniques:**	**Combined Linear and Constant Envelope Modulation Techniques**	**Spread Spectrum Modulation Techniques**
BPSK : Binary Phase Shift Keying	BFSK : Binary Frequency Shift Keying	MPSK: M-ary Phase Shift Keying	DS-SS : Direct Sequence Spread Spectrum
DPSK : Differential Phase Shift Keying	MSK : Minimum Shift Keying	QAM : M-ary Quadrature Amplitude Modulation	FH-SS : Frequency Hopped Spread Spectrum
QPSK : Quadrature Phase Shift Keying	GMSK : Gaussian Minimum Shift Keying	MFSK : M-ary Frequency Shift Keying	

Table II-1. Popular Digital Modulation Schemes.

2. M-ary Frequency Shift Keying Modulation Scheme

M-FSK (or M-ary FSK) transmits digital data by shifting the output frequency between M predetermined values (i.e., states). M-FSK is not particularly spectrally efficient, but offers advantages such as immunity to amplitude noise, bit rate higher than baud rate, and constant transmitter power [GRE00]. M-FSK requires less transmitted power for the same information rate than other digital modulation schemes do because it does not contain any AM components, as is the case for example for M-QAM. Thus, M-FSK allows transmitter power amplifiers to operate close to their saturation levels. In M-FSK modulation the M different frequencies on which the transmitted message is quantized are given by:

$$s_k(t) = g(t)\cos\left[\frac{\pi}{T}(n_c + k)t\right], \quad 0 \le t \le T, k = 1,2,\ldots,M, \qquad (2.1)$$

where g(t) is the signal pulse shape, T is the symbol duration, and $f_c = n_c/2T$ is the carrier frequency for a fixed integer n_c [WIL99].

3. M-phase Shift Keying Modulation Scheme

The most common form of modulation in digital communication is M-ary phase shift keying (M-PSK). With this method, symbols are distinguished from one another by the phase changes, while the amplitude remains the same. A digital symbol is represented by one of M different phase states of a sinusoidal carrier. The typical M-PSK waveform is given by:

$$s_k(t) = g(t) \cdot \cos(2\pi f_c t + \frac{2\pi}{M}(k-1)),$$

$$0 \le t \le T \ , \ k = 1,2,\ldots,M, \tag{2.2}$$

where g(t) is the signal pulse shape, M is the number of the possible phases of the carrier, T is the symbol duration and f_c is the carrier frequency [PRO95, pp.177,eq.4.3-11]. Figure II-2 plots the constellations for 2-PSK, 4-PSK, and 8-PSK modulation schemes.

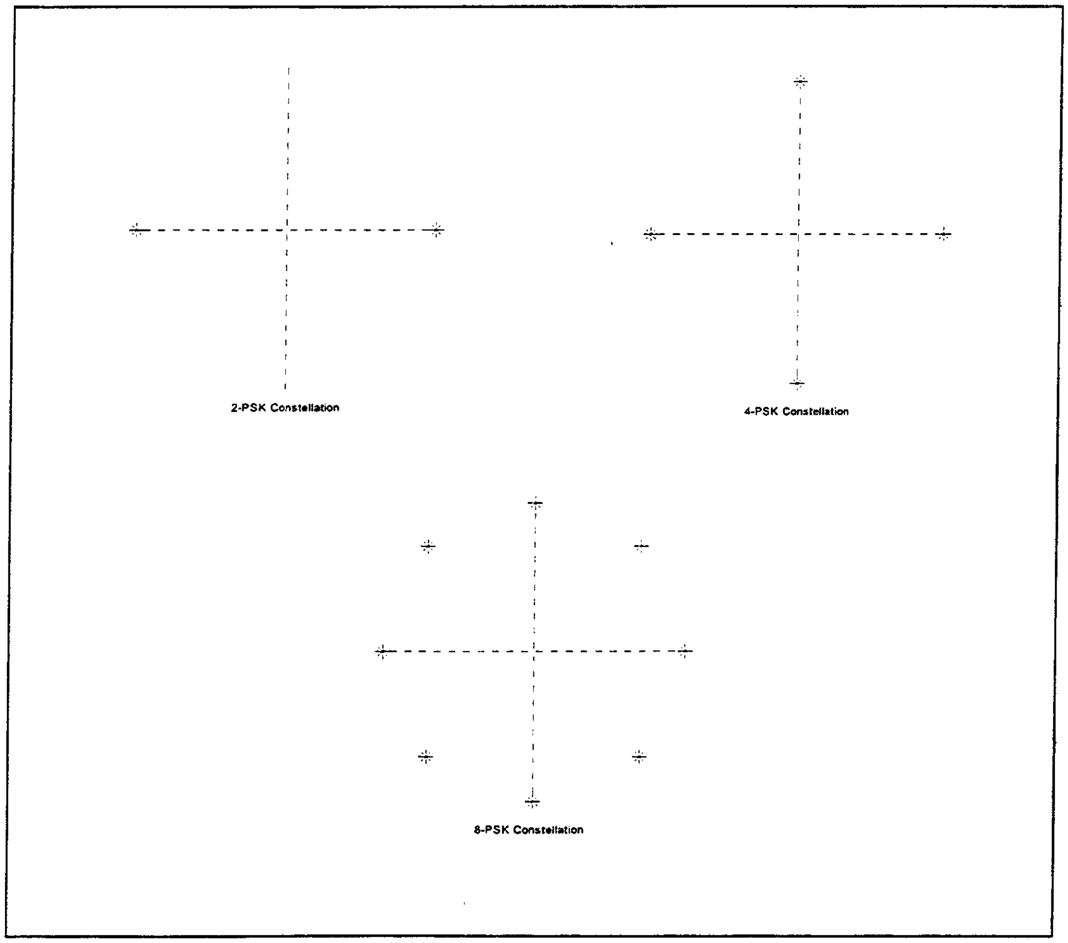


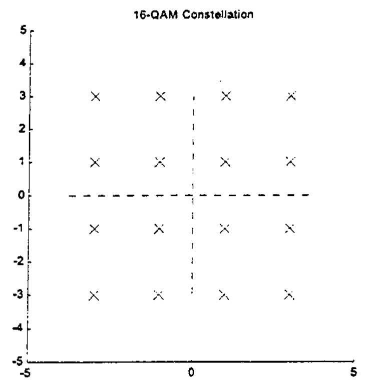
Figure II-2. 2-PSK, 4-PSK, 8-PSK constellations.

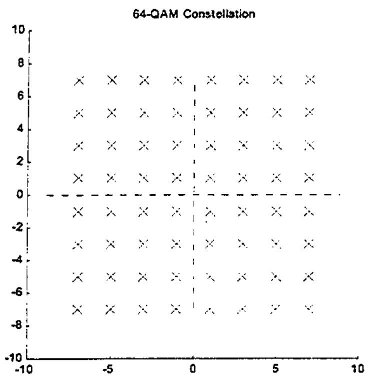
4. M-QAM Modulation Scheme

QAM is designed to transmit two separate signals independently with the same carrier frequency by using two quadrature carriers $\cos(2\pi f_c t)$ and $\sin(2\pi f_c t)$. These two separate modulated signals are then added and transmitted. This structure of QAM allows for M discrete amplitude levels (M-QAM), and thus permits a symbol to contain more than one bit of information. The general form for a M-QAM signal is given by:

$$s_k(t) = a_k g(t)\cos(2\pi f_c t) - b_k g(t)\sin(2\pi f_c t),$$
$$0 \le t \le T \ , \ \ k = 1,2,\ldots,M, \tag{2.3}$$

where g(t) is the signal pulse shape, and a_k and b_k are the information-bearing signal amplitudes of the quadrature carriers [PRO95, pp.179, eq.4-3-19]. 16-QAM, 64-QAM and 256-QAM constellations are shown in Figure II-3.





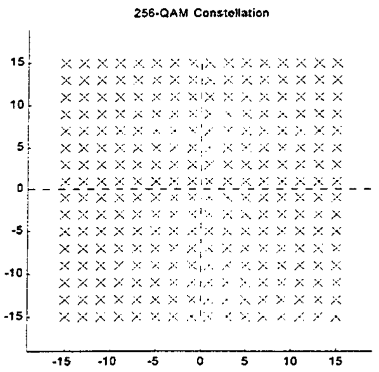


Figure II-3. 16-QAM, 64-QAM and 256-QAM constellations.

QAM is standardized in terms of the number M of discrete levels number which is chosen to be a power of 2 so that each symbol can be represented by a specific number of bits. For example, in 256-QAM, the number of discrete levels $M=256=2^8$, and every symbol is encoded with 8 bits. Therefore, higher order M-QAM schemes are much more spectrally efficient, being however, quite susceptible to noise and fading. As a result, higher order M-QAM schemes are more often used nowadays in cable transmission systems rather than wireless systems where transmission degradation may be worse.

5. Pulse Shaping Filters

Most digital communication signals, especially wireless ones, have limited bandwidth available to allow for simultaneous transmission of several messages. As a result, the modulated signal is passed through a transmission filter prior to transmission. In addition, transmission channels are usually band-limited, which leads to inter-symbol interference (ISI) in the transmitted signal. Therefore, it is important that the transmission filter be designed not to further increase the amount of ISI in the transmitted signal. Raised cosine filters are designed so that the ISI introduced by the filter band-limited structure is equal to zero when sampled at correct sample points [EVA00]. The raised cosine impulse response and frequency response are respectively given by:

$$x(t) = \sin c(\frac{\pi t}{2}) \cdot \left[\frac{\cos(\frac{\pi \beta t}{T})}{1-(\frac{2\beta t}{T})^2} \right], \tag{2.4}$$

$$X(f) = \begin{cases} T & 0 \le f \le \frac{1-\beta}{2T} \\ \frac{T}{2}\left\{1-\cos\left[\frac{\pi T}{\beta}(f-\frac{1-\beta}{2T})\right]\right\} & \frac{1-\beta}{2T} \le f \le \frac{1+\beta}{2T} \\ 0 & f > \frac{1+\beta}{2T}, \end{cases} \tag{2.5}$$

where T is the symbol period and $\beta \in [0,1]$ is called the roll-off factor, or excess bandwidth. Figure II-4 shows the raised cosine filter spectral characteristics and the corresponding pulses for β=0, 0.5 and 1.

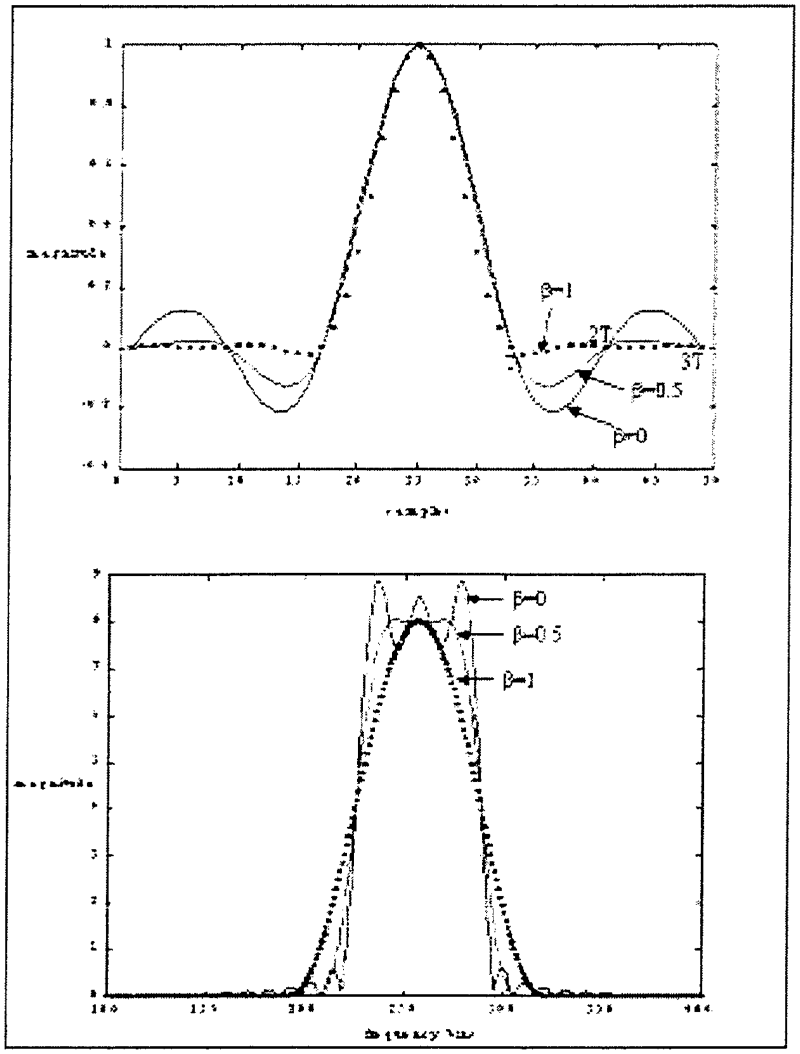


Figure II-4. Raised Cosine Impulse Response and Spectrum.

THIS PAGE INTENTIONALLY LEFT BLANK

III. WIRELESS TRANSMISSION CHANNELS

Chapter II briefly discussed digital modulation fundamentals. Chapter III, considers issues relevant to radio-wave propagation.

A. DESCRIPTION

Wireless environments have some inherent peculiarities concerning the signal transmission. There is a certain degree of randomness incorporating all those natural and sometimes unpredictable factors that might exist, such as geographical terrain, atmospheric conditions, temperature, other transmissions, even relative speed between transmitter and receiver. There are two main types of approaches to model a wireless transmission channel. A possible approach is to use statistical methods based on propagation laws. The other one is to apply empirical methods, by taking direct measurements in different typical wireless environments. However, no matter which philosophy is adopted, two main channel model categories exist; small scale fading and large scale fading transmission channel models. Both model types are considered next.

1. Small Scale Fading

Two different kinds of small scale fading exist in wireless propagation. Fading due to the "time spread", and fading due to the "doppler shift".

a) "Time Spread" Fading

In a real world situation transmitted radio signals follow different paths due to multipath reflection. Different propagation paths result in different delay times for each path, and therefore a time spread between the first and the last ray can be measured. This phenomenon may cause intersymbol interference (ISI), as a delayed symbol overlaps with another one that follows. A channel subject to time spread looks like a series of pulses in the time domain, as shown in Figure III-1.

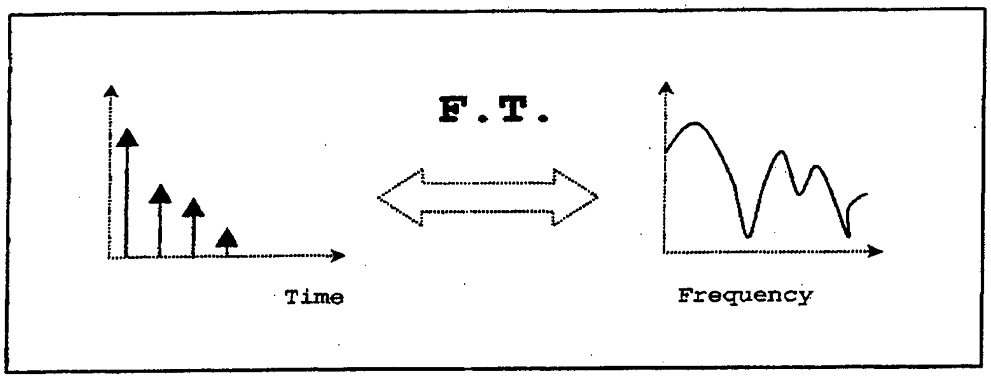


Figure III-1. Time Spread Effect in Small Scale Fading.

b) *"Doppler Shift" Fading*

Whenever there is a relative speed between a transmitter and a receiver, the carrier frequency at the receiver is shifted from that at the transmitter due to the Doppler effect. This frequency shift is given by:

$$f_d = \frac{v_{relative}}{c} f_c \, , \tag{3.1}$$

where $v_{relative}$ is the relative speed between the transmitter and the receiver, c is the speed of light and f_c is the carrier frequency [RAP99, p.165]. As a result, a broadening of the signal spectrum is observed. For the case of a sine wave, this frequency dispersion can be characterized by the U-shaped power spectrum given in Equation 3.2 and shown in Figure III-2 [HAA96]. The frequency range where the power spectrum is nonzero defines the Doppler spread f_d.

$$S_c(f) = \begin{cases} \dfrac{2}{\pi f_d} \cdot \dfrac{1}{\sqrt{1 - 4\dfrac{f^2}{f_d^2}}} & |f| \leq \dfrac{f_d}{2} \\ 0 & |f| > \dfrac{f_d}{2}. \end{cases} \tag{3.2}$$

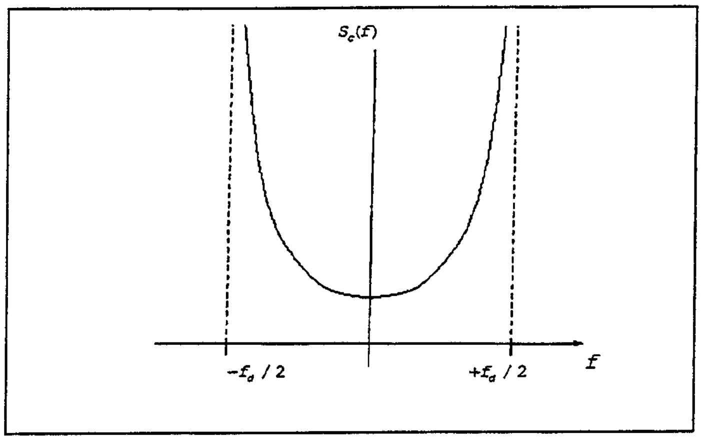


Figure III-2. PSD of a Sine Wave with a Doppler Shift.

2. Free Space Path Loss

Free space path loss concerns the attenuation of the signal strength with the distance from the transmitting source. In free space propagation the relationship between the transmitted and the received power is given by:

$$P_r = P_t \cdot G_t \cdot G_R \cdot \left(\frac{\lambda}{4\pi d}\right)^2 , \qquad (3.3)$$

where P_r is the received power, P_t is the transmitted power, G_t is the transmission gain and G_R is the reception gain. Equation 3.3 shows that the strength of the received power of a radiowave falls off as the inverse square of the distance between the transmitter and the receiver.

B. TRANSMISSION CHANNEL MODELING

The implementation of a realistic transmission channel is essential for the performance evaluation of every signal classification method. Such a specification is essential as the transmission channel can severely affect the transmitted signal either by increasing the inter-symbol interference or by lowering the effective SNR level. This study will solely discuss small scale fading situations, that is, time spread fading and Doppler shift fading.

1. Additive White Gaussian Noise Channel Model

The most common textbook channel is the additive white Gaussian noise (AWGN) channel, where the desired signal is degraded by thermal noise associated with the physical channel itself and/or other hardware used in the link. The AWGN-only channel is close to reality in some cases, such as space communications and forward path cable television (CATV).

2. Raised Cosine Channel Model

Rappaport [RAP, p.146, Eq. 4.12] introduces the impulse response of a multipath channel when receiver and transmitter are not in relative motion. Ideally this impulse response consists of a series of delta functions with decaying magnitudes (Figure III-1).

For all practical purposes these delta functions may be replaced with raised cosine functions that can be easily implemented in the real world. Time-spread between the multiple ray-paths and attenuation due to multipath propagation will be the two parameters that this channel takes into account. The analytic expression for the three-ray channel transfer function is given by:

$$h(t)=\operatorname{sinc}\left(\frac{t}{T}\right)\cdot\frac{\cos\left(\frac{\pi\beta t}{T}\right)}{1-\frac{4\beta^2t^2}{T^2}}+m_1\cdot\operatorname{sinc}\left(\frac{t-d_1}{T}\right)\cdot\frac{\cos\left(\frac{\pi\beta(t-d_1)}{T}\right)}{1-\frac{4\beta^2(t-d_1)^2}{T^2}}+m_2\cdot\operatorname{sinc}\left(\frac{t-d_2}{T}\right)\cdot\frac{\cos\left(\frac{\pi\beta(t-d_2)}{T}\right)}{1-\frac{4\beta^2(t-d_2)^2}{T^2}}, \qquad (3.4)$$

where T is the symbol duration, β is the filter's roll-off factor, m_1 is the attenuation of the 2nd ray, d_1 is the time difference between the 1st and the 2nd ray, m_2 is the attenuation of the 3rd ray and d_2 is the time difference between the 1st and the 3rd ray. Figure III-2 plots the impulse response and the spectrum of a 3-ray raised cosine channel model.

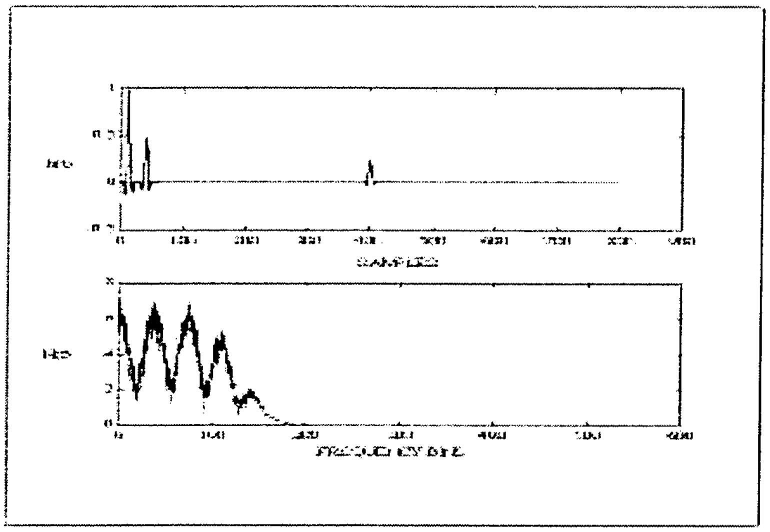


Figure III-3. Impulse response and spectrum for the 3-ray raised cosine channel model, $T = 8\times10^{-6}\sec, d_1 = 20\times10^{-6}\sec, d_2 = 200\times10^{-6}\sec$, β=0.35, $m_1 = -3dB$, $m_2 = -6dB$.

3. Rayleigh Channel Model

Rayleigh fading distribution is often used in wireless mobile communications to describe the statistical time varying nature of the received envelope of a flat fading signal, that is, a signal that has all ray paths attenuated uniformly. This means that there is no line of sight path between the transmitter and the receiver [LAU94]. This model may take into account the fact that the transmitter and the receiver might be in a relative motion, therefore time spread and Doppler shift may also be considered. The generic discrete expression of the received signal in a Rayleigh channel environment is given by:

$$r_k = \alpha_k s_k + n_k \ , \tag{3.5}$$

where α_k is a Rayleigh random variable, s_k is the signal sequence and n_k is noise. The envelope of a Rayleigh faded signal is shown in Figure III-4 [RAP99, pp. 173, Figure 4.15]. Deep fades occur when multipath components cancel one another. For the case where there are two principal components, this occurs when the difference in path lengths is multiple of half a wavelength. This is the cause of selective fading when the signal has finite bandwidth.

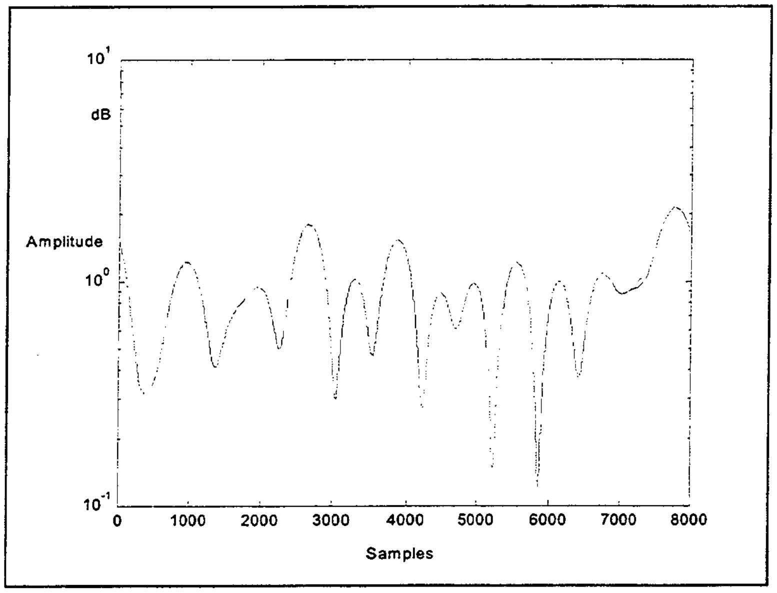


Figure III-4. Envelope of a Rayleigh faded signal, $f_c = 900 Mhz$, receiver speed=120Km/hr [RAP99, Figure 4.15].

The most popular model for simulating a Rayleigh fading signal is Clarke's model [RAP99, pp. 177-185]. This model assumes a fixed transmitter and a moving omni-directional receiver. Clarke showed that the power spectral density S(f) of the electric field in a Rayleigh fading environment, is generally given by Equation 3.6 [RAP, p.180, Eq. 4.76]:

$$S(f) = \frac{A[p(a)G(a) + p(-a)G(-a)]}{f_d \sqrt{1 - \left(\frac{f - f_c}{f_d} \right)^2}} \tag{3.6}$$

where f_d is the Doppler shift due to receiver's motion, f_c is the carrier frequency, A is the average received power with respect to an isotropic antenna, G(α) is the azimuthal gain pattern of the mobile antenna and p(α) is the received power within an angle α.

4. Ricean Channel Model

For Ricean fading there is a strong, constant component to the signal, in addition to the multiple random components of Rayleigh fading, due to multipath propagation [RAP99, pp. 174-176]. Ricean fading is typical in line-of-sight situations, where there is a direct path between transmitter and receiver, as well as reflecting or scattering phenomena. The Ricean case is often considered a characteristic of short-term indoor propagation, while the Rayleigh model fits well with outdoor, short-term propagation.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. INTRODUCTION TO SIGNAL EQUALIZATION

Chapter III presented an overview of the effect of the wireless environment on the transmitted signal. In real world situations the transmission channel is a critical factor that may cause unrecoverable distortions on the signal, especially in higher order digital modulations, such as in 256-QAM, where the effect of a propagation channel may corrupt the signal constellation even at high SNR levels. Figure IV-1 shows a 256-QAM sequence constellation obtained for SNR equal to 40dB at the transmitter. Figure IV-2 presents the constellation obtained by passing this 256-QAM signal through a severe urban area channel model [Appendix C, channel 11]. To compensate for this distortion, modern receivers use signal equalization extensively, in an attempt to undo the effects of the propagation channel. This chapter will discuss two types of signal equalization: the Constant Modulus Algorithm - Fractionally Spaced Equalizer (CMA-FSE) blind equalization method and the Alphabet Matched Algorithm (AMA) equalization method.

A. THE CMA-FSE ALGORITHM

The constant modulus algorithm with fractionally spaced equalizer (CMA-FSE) belongs to a category of equalization methods called blind equalization methods which are designed to undo the channel effect without any knowledge of the channel itself. The

CMA-FSE is the integration of two different parts: the constant modulus algorithm (CMA) and the fractional spaced equalizer (FSE).

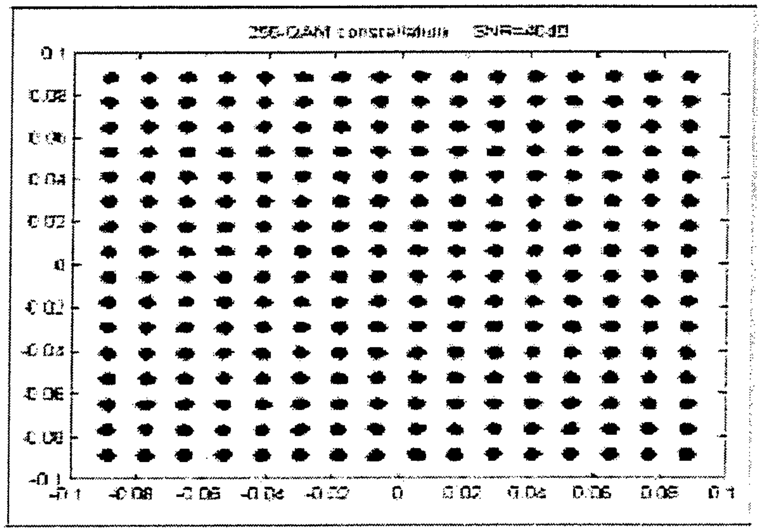


Figure IV-1. Ideal 256-QAM constellation; no propagation channel effect.

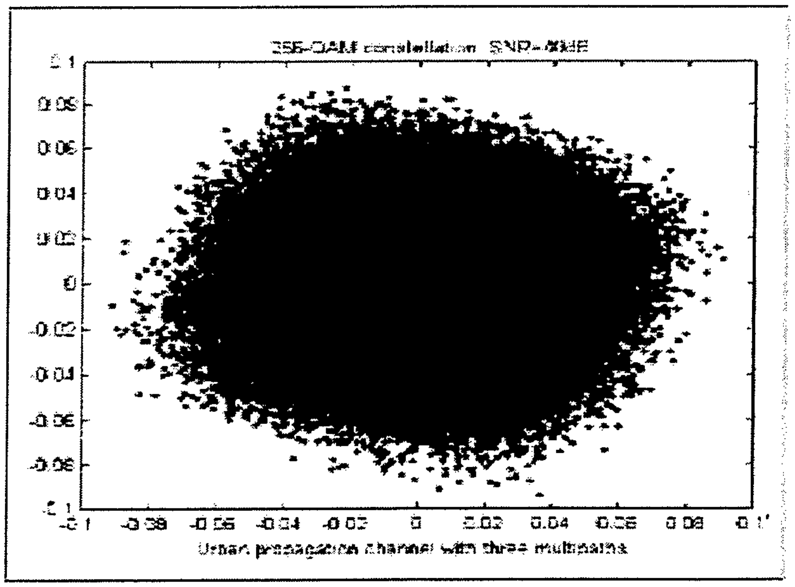


Figure IV-2. 256-QAM constellation; after signal transmission through non-linear channel 11. Channel specifications given in Appendix C.

1. Constant Modulus Algorithm

The Constant Modulus Algorithm (CMA) is a stochastic gradient algorithm, designed to force the equalizer weights to keep a constant envelope on the received signal [HAY96, pp. 365-372, RAP99, pp.304]. Thus, it is designed for problems where the signal of interest has a constant envelope property. However, extensive simulations have shown that it still can be used in amplitude-phase modulation types with success, when the number of states is low, and is routinely used in today's applications. As a result, the CMA is expected to have better performance for M-FSK and M-PSK rather than M-QAM types. The CMA cost function is given by:

$$J(n) = E\left\{\left(\left|s(n)\right|^2 - \gamma\right)^2\right\}, \tag{4.1}$$

where s(n) is the signal to equalize and γ is a positive real constant called the "dispersion constant" defined by:

$$\gamma = \frac{E_{s,4,4}}{E_{s,2,2}}, \tag{4.2}$$

where $E_{S,4,4}$ and $E_{S,2,2}$ are the 4th and 2nd order moments respectively [CJJ00]. These moments are described further in Chapter V. The cost function J(n) is minimized iteratively using a gradient-based algorithm with update equation:

$$\underline{h}(n+1) = \underline{h}(n) - \mu \nabla J(n), \qquad (4.3)$$

where $\underline{h}$ is the tap-weight vector and μ is the step-size parameter [HAY96, pp. 794-795].

2. Fractional Spaced Equalizer (FSE)

In any standard CMA equalization system, the coefficient taps are baud-spaced that is, the sampling frequency of the equalizer is the baud frequency of the received signal. However, it is often desired to use an equalizer with taps spaced at a fraction of the data symbol period T, or sampled at a multiple of the symbol rate. This configuration gives the extra degrees of freedom to perform additional filtering operations such as matched filtering and adjustment of sampling phase [HAJ99]. Such a scheme is called fractional spaced equalization (FSE). In a fractional spaced equalizer, the channel model is sampled usually at twice the symbol rate and the equalizer output is evaluated only at T-spaced intervals to obtain the equalized signal.

3. CMA-FSE Scheme

The implementation of a fractional spaced equalizer using the constant modulus criterion combines the advantages of both concepts into one system. This system is shown in Figure IV-3. The propagation channel is assumed to be linear and time invariant. Therefore, the channel C is modeled with a time-invariant finite impulse response (FIR) filter with coefficients $\underline{c} = [c_0, c_1, \ldots c_{Q-1}]^T$. The equalizer is also described by a N-coefficient vector $\underline{f} = [f_0, f_1, \ldots, f_{N-1}]^T$ and the overall system response is described by the P-coefficient vector $\underline{h} = [h_0, h_1, \ldots h_{P-1}]^T$. The filtering operation performed by the equalizer can be viewed as the convolution of the sampled received sequence with the equalizer coefficients. Therefore, the overall system response is $\underline{h} = \mathbf{C} \cdot \underline{f}$, where $\mathbf{C}$ is the $P \times N$ channel convolution matrix given by Equation (4.4) below [JAO98, pp. 1930, Equation 5].

$$\mathbf{C} = \underbrace{\begin{bmatrix} c_0 & & & & \\ c_1 & c_0 & & & \\ c_2 & c_1 & c_0 & \ddots & \\ \vdots & c_2 & c_1 & \ddots & \\ c_{Q-1} & \vdots & c_2 & \ddots & c_0 \\ & c_{Q-1} & \vdots & \ddots & c_1 \\ & & c_{Q-1} & \ddots & c_2 \\ & & & \ddots & \vdots \\ & & & & c_{Q-1} \end{bmatrix}}_{P \times N} \qquad (4.4)$$

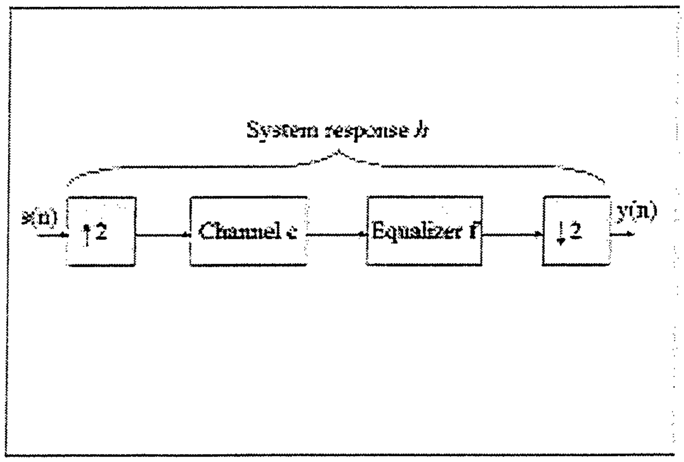


Figure IV-3. CMA-FSE Implementation Block Diagram.

4. Example

The CMA-FSE algorithm is tested on 4-PSK, 16-QAM, 64-QAM and 256-QAM modulation type signals where the SNR is set to 40dB for all cases. The purpose of this test is to find the limits of the highest constellation order that the CMA-FSE algorithm is able to clear. The CMA-FSE algorithm is implemented in MATLAB. The code was developed by researchers at the Blind Equalization Research group, Cornell University, and is presented in Appendix D. Figure IV-4 shows the impulse response of the propagation channel that the CMA-FSE scheme attempts to undo the effect of. This channel is a 2-path channel and is a typical example of a rural area environment.

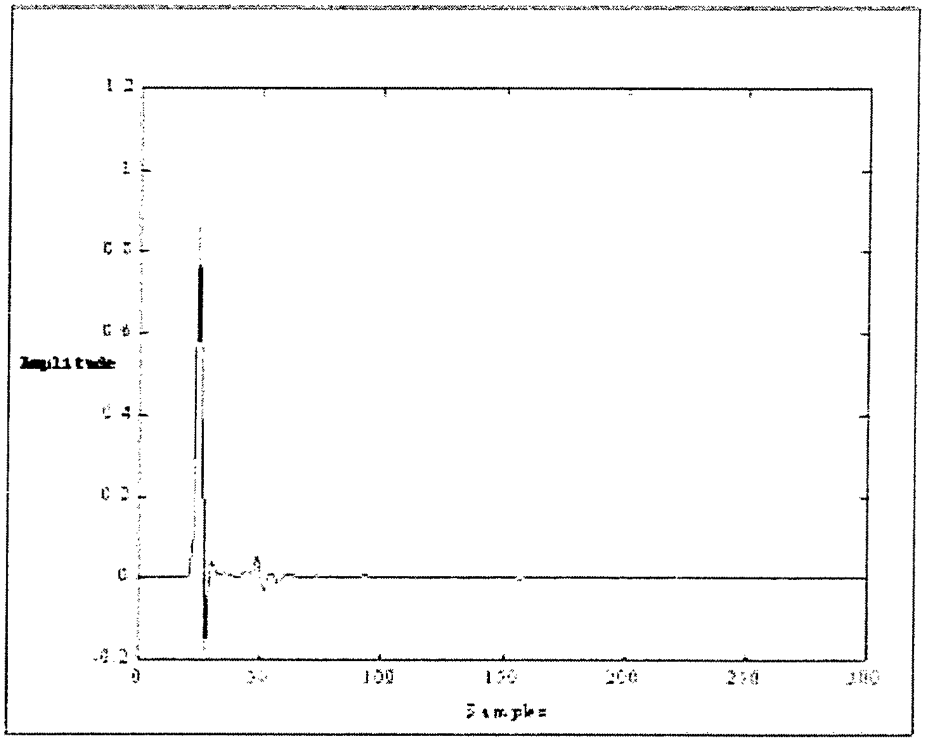


Figure IV-4. Impulse response of a rural area propagation channel.

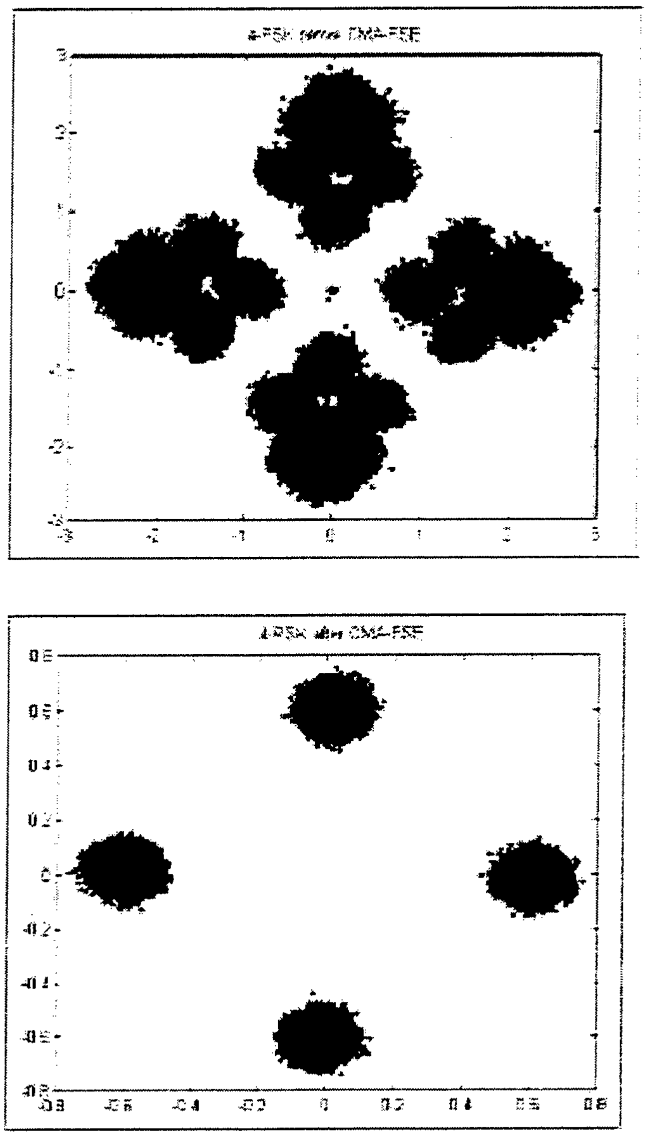


Figure IV-5. 4-PSK constellations; before and after applying the CMA-FSE algorithm.

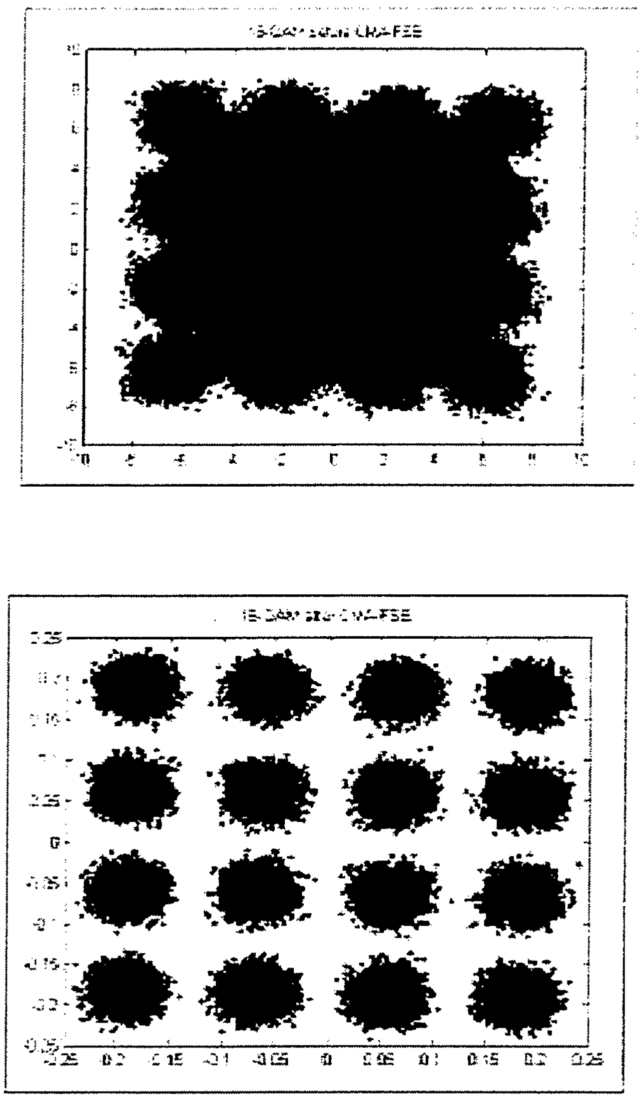


Figure IV-6. 16-QAM constellations; before and after applying the CMA-FSE algorithm.

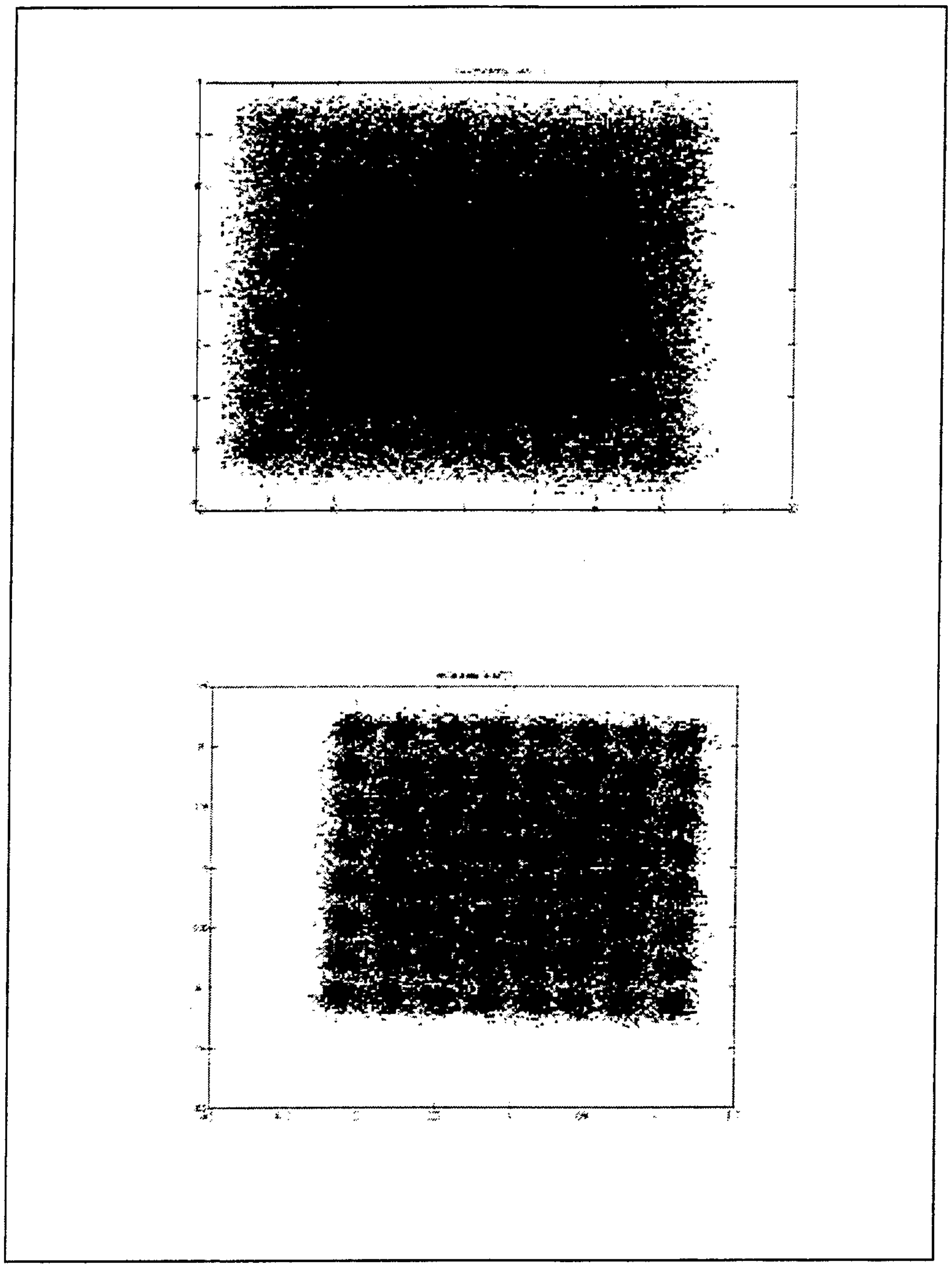


Figure IV-7. 64-QAM constellations; before and after applying the CMA-FSE algorithm.

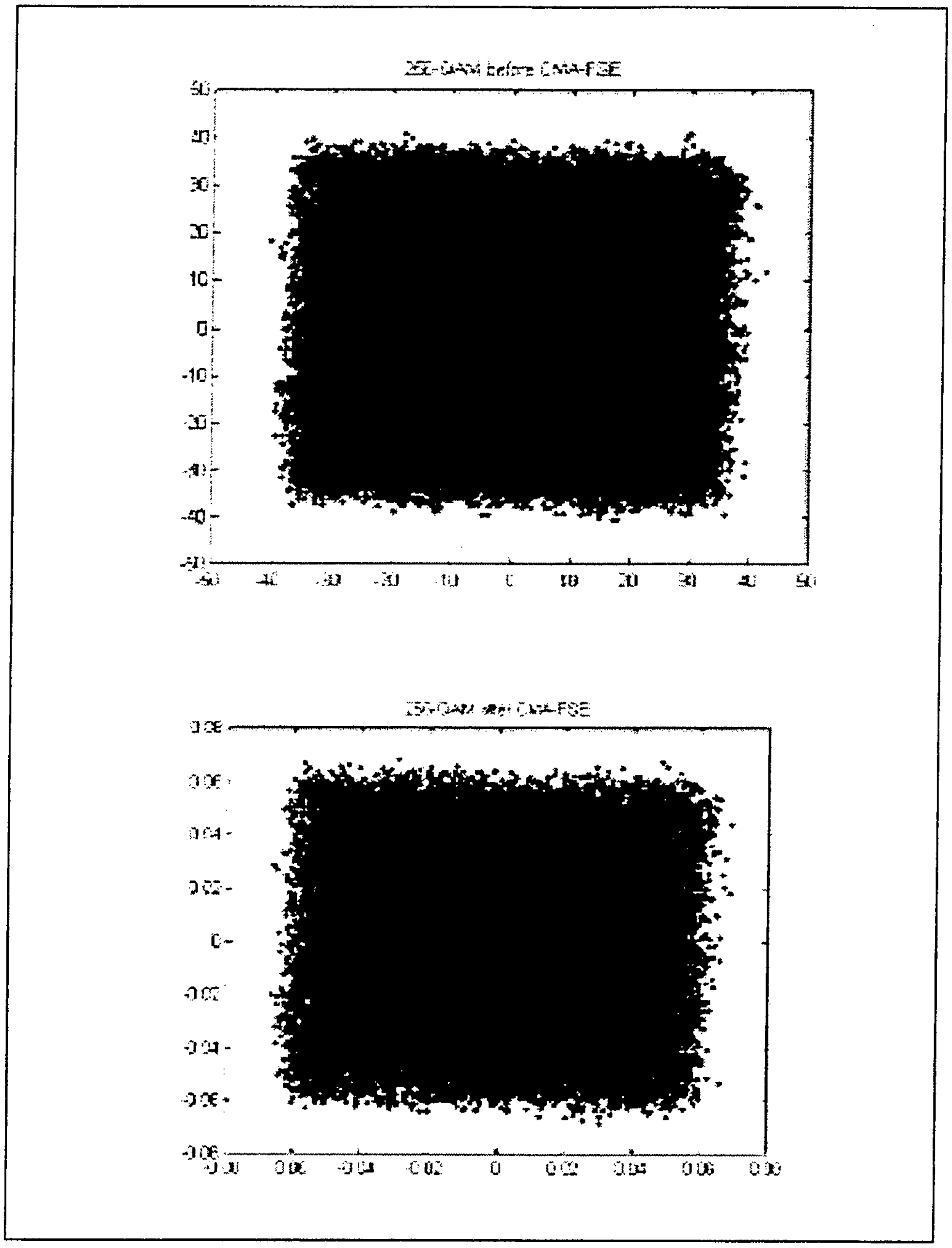


Figure IV-8. 256-QAM constellations; before and after applying the CMA-FSE algorithm.

Simulations show that the CMA-FSE implementation cancels the channel effect almost perfectly for modulations up to 16-QAM, as illustrated in Figures IV-5 and IV-6. Performances degrade for higher constellations. Figure IV-7 shows that the constellation type is still recognizable for 64-QAM, but Figure IV-8 indicates that CMA-FSE fails for 256-QAM. This is to be expected that this scheme was designed for constant magnitude modulations and not for QAM schemes, especially of higher order.

B. THE ALPHABET MATCHED ALGORITHM (AMA)

Applying the CMA for blind equalization is an efficient way to cope with QAM signals with relatively low order constellations. However, a different type of processing is needed to recover QAM signals with high constellation types. A possible alternative is to implement a non-blind approach which takes advantage of the specific information contained in a given signal type, such as constellation centers for example. Such an approach has been considered recently by [BSC98] and [BAR00] and will be discussed next.

1. Introduction

The Alphabet Matched Algorithm (AMA) is an equalization scheme that uses a-priori knowledge of the constellation centers for QAM signals with a specific number of states M. This approach was first reported by [BSC98] for M-QAM of low constellation orders [BAR00]. Barbarossa et. al. modified the original AMA to make it more robust in high constellation environments [BAR00]. The AMA implementation consists of a bank of FIR equalizers where each one is matched to a specific constellation type, as shown in Figure IV-9. The equalizer that achieves the smallest cost function after convergence indicates the modulation type [BSC98].

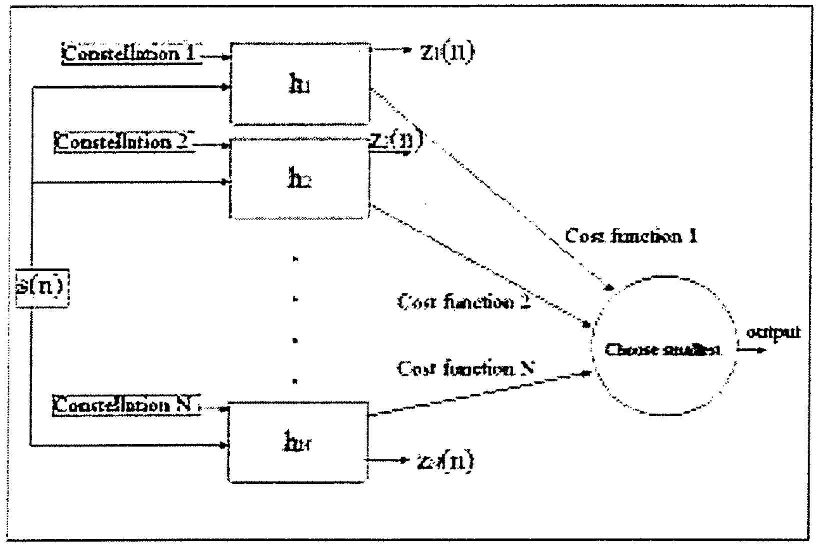


Figure IV-9. AMA classifier.

Let us examine a single branch of Figure IV-9 only, as similar findings hold for the others. Assume the L-tap FIR equalizer weight vector is denoted by:

$$h(n) = \left[h_0(n), \ldots, h_{L-1}(n)\right]. \tag{4.5}$$

Applying the equalizer filter to the input signal sequence s(n) leads to the equalizer output z(n):

$$z(n)=\sum_{l=0}^{L-1} h_l(n)\cdot s(n-l). \tag{4.6}$$

The basic difference between the CMA and the AMA implementations lies in the definition of the cost function $J_k(n)$ associated for the kth constellation defined as:

$$J_k(n)=E\left\{1-\sum_{i=1}^{M} e^{-|z(n)-c_k(i)|^2 / 2\sigma^2}\right\}, \tag{4.7}$$

where M represents the total number of centroids for the kth constellation, z(n) is the output of the equalizer, $c_k(i)$ is the ith centroid of the kth constellation, and σ is a constant chosen so that:

$$e^{-|c(l)-c(i)|^2/2\sigma^2} \approx 0, \qquad \forall l \neq i. \tag{4.8}$$

Basically, Equation (4.8) determines the allowed distance between the centroids and the equalizer output. The smaller the value of σ, the bigger the penalty of the cost function on the equalizer output. Figure IV-10 shows the AMA cost function obtained for a 64-QAM constellation modulation type.

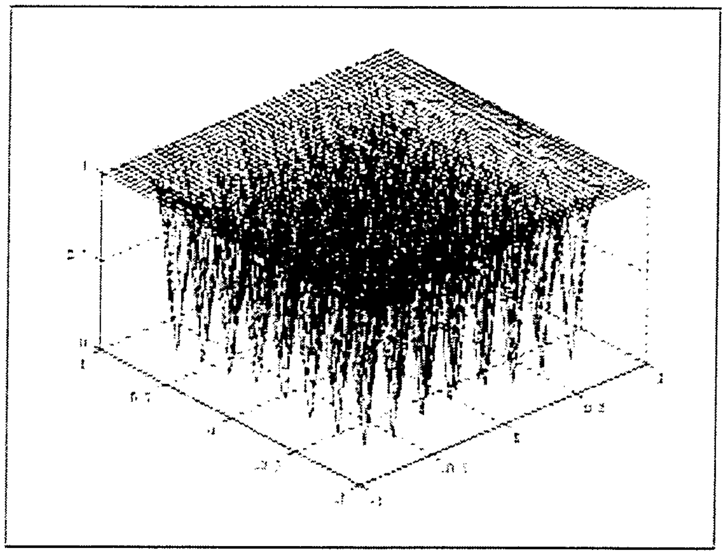


Figure IV-10. AMA cost function for 64-QAM with σ=0.05.

As before, the cost function J_k is minimized iteratively using a gradient descent algorithm. The update equation for the filter coefficients is given by:

$$\underline{h}_k(n+1)=\underline{h}_k(n)-\mu\nabla J_k[z(n)],\quad k=1,..P, \tag{4.9}$$

where μ is the step size, and P is the total number of QAM constellation considered. The gradient derivation is presented in Appendix A, and the final expression is given by:

$$\nabla J_k(\underline{h}) = \sum_{i=1}^{M} \left[e^{\frac{\left|\underline{h}^T \underline{s} - c_k(i)\right|^2}{2\sigma^2}} \cdot (\underline{h}^T \underline{s} - c_k(i))^* \cdot \underline{s} \right], \tag{4.10}$$

where $\underline{s} = [s(n), s(n-1), \ldots s(n-L)]^t$ is a portion of the input signal with length equal to the length of the filter equalizer.

2. Example

The AMA algorithm was tested on 16-QAM, 64-QAM and 256-QAM modulation signals with a SNR level of 40dB. Each signal was passed through the same propagation channel, as in the earlier CMA-FSE simulations considered in Section A. Next, the CMA-FSE algorithm was applied to the resulting transmitted signal to provide a good initialization to the AMA. Such a two-step process was followed as results have shown the AMA has good local convergence but needs good initialization [BAR00, p. 177]. Note that the CMA is known to have good global convergence properties when the symbol set is close to being constant modulus, even when the initialization is poor. Therefore, cascading both schemes should allow for a more robust modulation type decision. As a result, the AMA algorithm is initialized when the CMA-FSE converges. Figures IV-10, IV-11 and IV-12 show the simulation results. The MATLAB implementation of the AMA algorithm is presented in Appendix D.

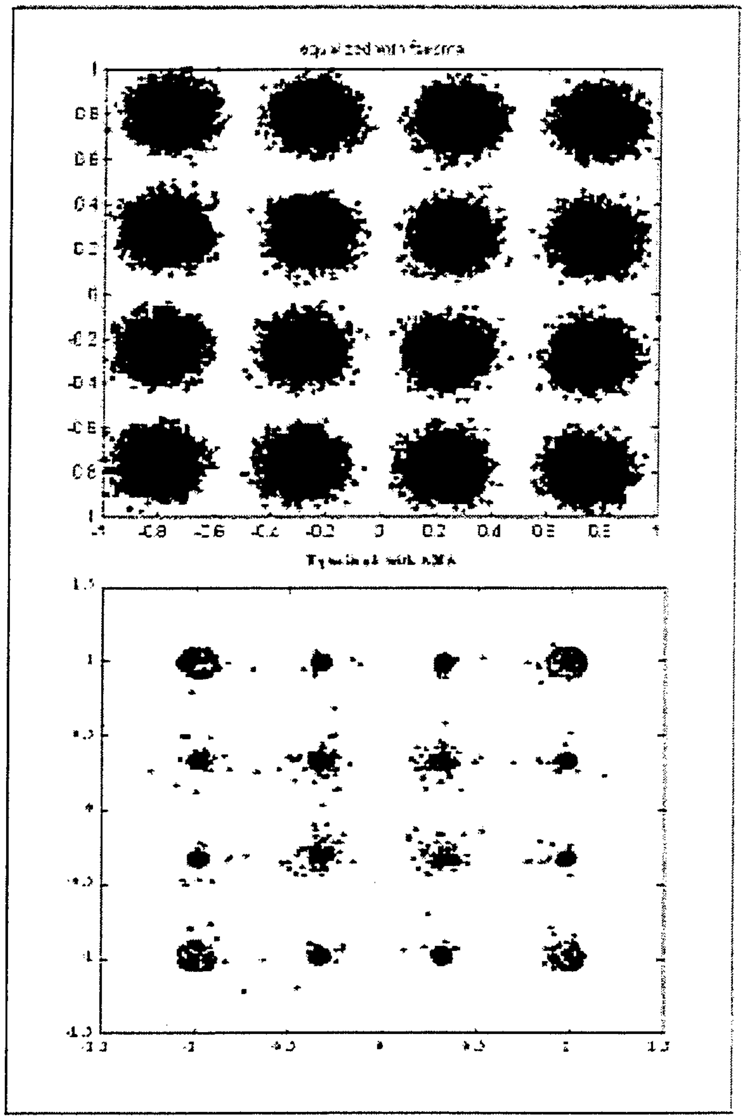


Figure IV-11. 16-QAM constellations; before and after applying the AMA algorithm. SNR=40dB, step size μ=0.01, σ=0.174, 2000 samples, 21 equalizer taps.

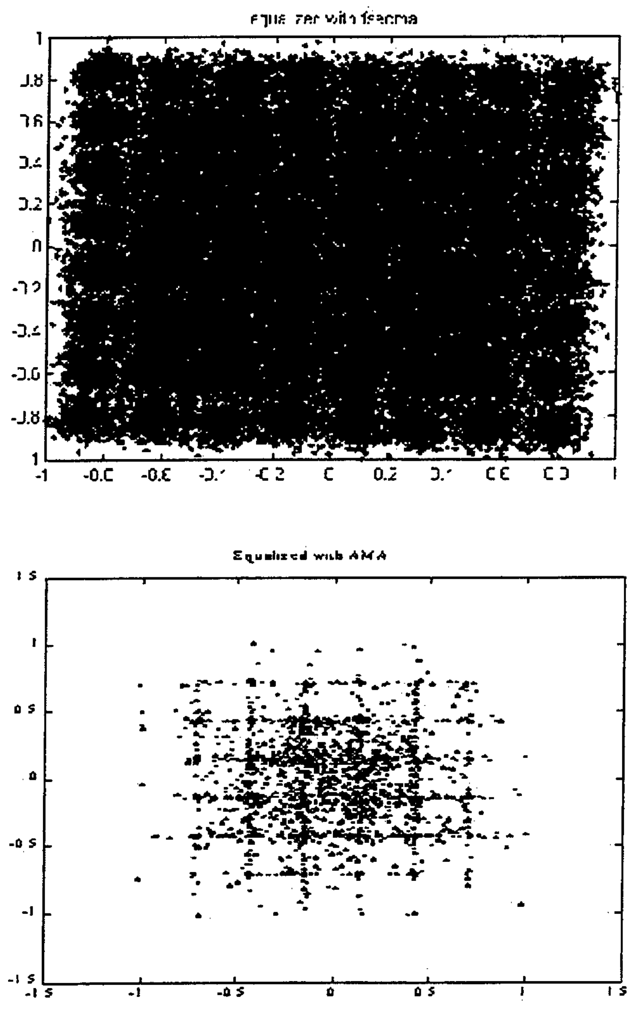


Figure IV-12. 64-QAM constellations; before and after applying the AMA algorithm. SNR=40dB, step size μ=0.01, σ=0.1174, 2000 samples, 21 equalizer taps.

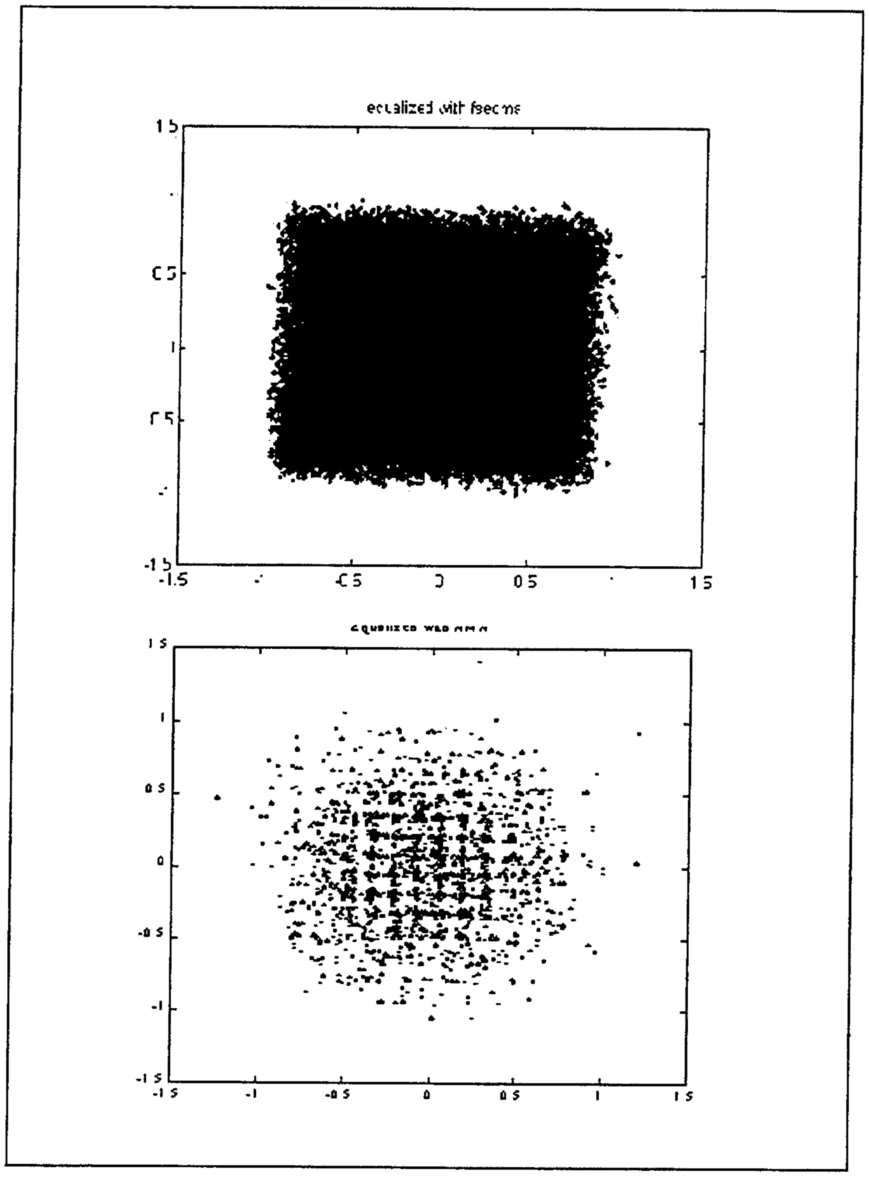


Figure IV-13. 256-QAM constellations; before and after applying the AMA algorithm. SNR=40dB, step size μ=0.01, σ=0.05, 2000 samples, 21 equalizer taps.

Results show the AMA gives very good results in the first two cases. Even in 256-QAM, where the CMA-FSE has no effect, the AMA algorithm recovers a portion of the constellation. Simulations showed that the key to the AMA algorithm convergence is the value of σ and the step size. Recall that the parameter σ controls the sharpness of the cost function peaks. Simulations showed that some samples of the signal can potentially be assigned to the wrong centroid when σ is selected too large, due to overlap of the cost function nulls (Figure IV-10). In addition, the AMA may not converge, when the step size is chosen too large or too small.

V. MOMENTS AND CUMULANTS

Chapter III gave a brief description of the major propagation channels and their resulting effects on the quality of the received sequence at the receiver. Chapter IV discussed two different equalization schemes designed to minimize channel distortions effects (CMA-FSE and AMA algorithms). This chapter focuses on identifying features that can be used to identify signals subjected to various types of distortion. As mentioned earlier in Section 1.C, higher-order statistics have been extensively used to extract unique signal features. Higher-order statistics is a field of statistical signal processing which makes use of additional information to that usually used in 'traditional' signal processing measures, such as the power spectrum and autocorrelation function. Advantages of higher order statistics include the ability to identify non-Gaussian processes and non-minimum phase systems, and to detect and characterize signal non-linear properties. Higher-order statistics lead to the definition of two directly related parameters: statistical moments and cumulants, which are described next.

A. MOMENTS

1. Definition

Probability distribution moments are a generalization of the concept of the expected value, and can be used to define the characteristics of a probability density function. Recall that the general expression for the i^{th} moment of a random variable is given by:

$$\mu_i = \int_{-\infty}^{\infty} (s-\mu)^i f(s)ds, \tag{5.1}$$

where μ is the mean of the random variable. The definition for the i^{th} moment for a finite length discrete signal is given by:

$$\mu_i = \sum_{k=1}^{N} (s_k - \mu)^i f(s_k), \tag{5.2}$$

where N is the data length. In this study signals are assumed to be zero mean. Thus Equation 5.2 becomes:

$$\mu_i = \sum_{k=1}^{N} {s_k}^i f(s_k). \tag{5.3}$$

Next, the auto-moment of the random variable may be defined as:

$$E_{s,p+q,p} = E\left[s^p (s^*)^q\right], \tag{5.4}$$

where p and q represent the number of the non conjugated terms and number of the conjugated terms, respectively, and p+q is called the moment order. For example, for p=2 and q=0, Equation 5.4 becomes:

$$E_{s,2,2} = E\left[s^2 (s^*)^0\right] = E\left[s^2\right] = \mu_2 = \sum_{k=1}^{N} s_k{}^2 f(s_k), \tag{5.5}$$

which is the second moment or the variance of the random variable. In a similar way, expressions for $E_{s,2,1}, E_{s,4,4}, E_{s,8,4}$, etc...may be easily derived. Note that the normalized moments $E_{s,3,3}$ and $E_{s,4,4}$ are called Skewness and Kurtosis respectively. Skewness is a measure of the symmetry of the pdf, whereas Kurtosis is the degree of peakedness (density of peaks) of the pdf.

2. Explicit Calculation of Major Moments

Selecting second or higher order moments has already proved to be promising to characterize communication signals, as they may be used to describe the shape of the pdf of a distribution completely [MAB97]. In a sense, the sequence of moments is analogous to the components of a Fourier sequence; the first few terms describe the general shape and the later terms add up to more detail. Therefore it is useful to derive expressions that give some commonly used higher order moments.

Assume a zero mean discrete base-band signal sequence of the form $s_k = a_k + j \cdot b_k$. Using the definition of the auto-moments (Equation 5.4), the expressions for moments of order 2, 4, 6 and 8 may be easily derived. Complete derivations are given in Appendix B, and the results are summarized in Table V-1.

ORDER 2	$E_{S,2,2}$	$E\left[a^2 - b^2\right]$
	$E_{S,2,1}$	$E\left[a^2 + b^2\right]$
ORDER 4	$E_{S,4,4}$	$E\left[a^4 + b^4 - 6a^2b^2\right]$
	$E_{S,4,3}$	$E\left[a^4 - b^4\right]$
	$E_{S,4,2}$	$E\left[a^4 + b^4 + 2a^2b^2\right]$
ORDER 6	$E_{S,6,6}$	$E\left[a^6 - b^6 + 15a^2b^4 - 15a^4b^2\right]$
	$E_{S,6,5}$	$E\left[a^6 + b^6 - 5a^2b^4 - 5a^4b^2\right]$
	$E_{S,6,4}$	$E\left[a^6 - b^6 - a^2b^4 + a^4b^2\right]$
	$E_{S,6,3}$	$E\left[a^6 + b^6 + 3a^2b^4 + 3a^4b^2\right]$
ORDER 8	$E_{S,8,8}$	$E\left[a^8 + b^8 - 28a^6b^2 + 70a^4b^4 - 28a^2b^6\right]$
	$E_{S,8,7}$	$E\left[a^8 - b^8 - 14a^6b^2 + 14a^2b^6\right]$
	$E_{S,8,6}$	$E\left[a^8 + b^8 - 4a^6b^2 - 10a^4b^4 - 4a^2b^6\right]$
	$E_{S,8,5}$	$E\left[a^8 - b^8 + 2a^6b^2 - 2a^2b^6\right]$
	$E_{S,8,4}$	$E\left[a^8 + b^8 + 4a^6b^2 + 6a^4b^4 + 4a^2b^6\right]$

Table V-1. Statistical moments; zero-mean sequence of the form $s_k = a_k + j \cdot b_k$

B. CUMULANTS

1. Definition

Consider a scalar zero mean random variable *s* with characteristic function:

$$\hat{f}(t) = E\left\{e^{its}\right\}. \quad (5.6)$$

Expanding the logarithm of the characteristic function as a Taylor series, one obtains:

$$\log \hat{f}(t) = k_1(it) + \frac{k_2(it)^2}{2} + \ldots + \frac{k_r(it)^r}{r!} + \ldots, \quad (5.7)$$

where the constant k_r are called the cumulants (of the distribution) of *s* [HYY00]. Note that the first three cumulants (for zero-mean variables) are identical to the first three moments:

$$\begin{aligned} k_1 &= E\{s\} \\ k_2 &= E\left\{s^2\right\} = E_{s,2,2} \\ k_3 &= E\left\{s^3\right\} = E_{s,3,3}. \end{aligned} \quad (5.8)$$

The symbolism for the nth order cumulant is similar to that of the nth order moment. More specifically:

$$C_{s,p+q,p} = Cum\left[\underbrace{s,\ldots,s}_{p \quad terms},\underbrace{s^*,\ldots,s^*}_{q \quad terms}\right]. \qquad (5.9)$$

2. Relation Between Cumulants and Moments

The n^{th} order cumulant is a function of the moments of orders up to (and including) n. Moments may be expressed in terms of cumulants as:

$$E[s_1\ldots s_n] = \sum_{\forall v} Cum\left[\left\{s_j\right\}_{j\in v_1}\right]\ldots Cum\left[\left\{s_j\right\}_{j\in v_q}\right], \qquad (5.10)$$

where the summation index is over all partitions $v=(v_1,\ldots,v_q)$ for the set of indexes $(1,\ldots,n)$, and q is the number of elements in a given partition. Cumulants may also be derived in terms of moments. The nth order cumulant of a discrete signal $s(n)$ is given by:

$$Cum[s_1,\ldots,s_n] = \sum_{\forall v}(-1)^{q-1}(q-1)!E\left[\prod_{j\in v_1} s_j\right]\ldots E\left[\prod_{j\in v_q} s_j\right], \qquad (5.11)$$

where the summation is being performed an all partitions $v=(v_1,\ldots,v_q)$ for the set of indexes $(1,\ldots,n)$. A simple application example for equation 5.11 is presented next.

a) Example

Assume *n=1*. In such a case, only one partition v_1 can be defined. Therefore, *q=1*, and equation (5.11) leads to:

$$Cum[s_1]=(-1)^{1-1}(1-1)!E[s_1]\Rightarrow Cum[s_1]=E[s_1]. \tag{5.12}$$

Assume *n=2*. In such a case, the available set of indexes is *1* and *2*, and two different types of partitioning may be obtained for that set. Thus, $v=(v_1,v_2)$. The partitions are:

- (1,2) with q=1,
- (1), (2) with q=2.

Therefore, equation (5.11) becomes:

$$\begin{aligned}&Cum[s_1,s_2]=(-1)^{1-1}(1-1)!E[s_1s_2]+(-1)^{2-1}(2-1)!E(s_1)E[s_2]\Rightarrow\\&Cum[s_1,s_2]=E[s_1s_2]-E(s_1)E[s_2].\end{aligned} \tag{5.13}$$

Finally, assume *n=3*. In such a case, the available set of indexes is *(1,2,3)*, and four different types of partitioning may be obtained for that set. Thus, $\nu = (\nu_1, \nu_2, \nu_3, \nu_4)$. These partitions are:

- (1,2,3,) leading to q=1,
- (1), (2,3) leading to q=2,
- 2, (1,3) leading to q=2,
- 3, (1,2) leading to q=2,
- (1), (2), (3) leading to q=3.

Therefore, Equation (5.11) becomes:

$$\begin{aligned}
Cum[s_1,s_2,s_3] = {} & (-1)^{1-1}(1-1)!E[s_1s_2s_3]+ \\
& +(-1)^{2-1}(2-1)!E[s_1]E[s_2s_3]+ \\
& +(-1)^{2-1}(2-1)!E[s_2]E[s_1s_3]+ \\
& +(-1)^{2-1}(2-1)!E[s_3]E[s_1s_2]+ \\
& +(-1)^{3-1}(3-1)!E[s_1]E[s_2]E[s_3] \Rightarrow \\
Cum[s_1,s_2,s_3] = {} & E[s_1s_2s_3]-E[s_1]E[s_2s_3]-E[s_2]E[s_1s_3]-E[s_3]E[s_1s_2]+ \\
& +2E[s_1]E[s_2]E[s_3].
\end{aligned} \tag{5.14}$$

Marchand computed similar cumulant expressions up to the 8th order [MAR98, pp. 173-174], and these are presented in Table V-2.

Order 2	$C_{s,2,2} = E_{s,2,2}$
	$C_{s,2,1} = E_{s,2,1}$
Order 4	$C_{s,4,4} = E_{s,4,4} - 3E_{s,2,2}^{2}$
	$C_{s,4,3} = E_{s,4,3} - 3E_{s,2,2}E_{s,2,1}$
	$C_{s,4,2} = E_{s,4,2} - E_{s,2,2}^{2} - 2E_{s,2,1}^{2}$
Order 6	$C_{s,6,6} = E_{s,6,6} - 15E_{s,2,2}E_{s,4,4} + 30E_{s,2,2}^{3}$
	$C_{s,6,5} = E_{s,6,5} - 10E_{s,2,2}E_{s,4,3} - 5E_{s,2,1}E_{s,4,4} + 30E_{s,2,2}^{2}E_{s,2,1}$
	$C_{s,6,4} = E_{s,6,4} - E_{s,2,2}E_{s,4,4} - 8E_{s,2,1}E_{s,4,3} - 6E_{s,2,2}E_{s,4,2} + 6E_{s,2,2}^{3}$ $+ 24E_{s,2,1}^{2}E_{s,2,2}$
	$C_{s,6,3} = E_{s,6,3} - 6E_{s,2,2}E_{s,4,3} - 9E_{s,2,1}E_{s,4,2} + 18E_{s,2,2}^{2}E_{s,2,1} + 12E_{s,2,1}^{3}$
Order 8	$C_{s,8,8} = E_{s,8,8} - 35E_{s,4,4}^{2} - 630E_{s,2,2}^{4} + 420E_{s,2,2}^{2}E_{s,4,4}$
	$C_{s,8,7} = E_{s,8,7} - 35E_{s,4,4}E_{s,4,3} - 630E_{s,2,2}^{3}E_{s,2,1} + 210E_{s,4,4}E_{s,2,2}E_{s,2,1}$ $+ 210E_{s,2,2}E_{s,4,3}$
	$C_{s,8,6} = E_{s,8,6} - 15E_{s,4,4}E_{s,4,2} - 20E_{s,4,3}^{2} + 30E_{s,4,4}E_{s,2,2}^{2} + 60E_{s,4,4}E_{s,2,1}^{2}$ $+ 240E_{s,4,3}E_{s,2,1}E_{s,2,2} + 90E_{s,4,2}E_{s,2,2}^{2} - 90E_{s,2,2}^{4} - 540E_{s,2,2}^{2}E_{s,2,1}^{2}$
	$C_{s,8,5} = E_{s,8,5} - 5E_{s,4,4}E_{s,4,3} - 30E_{s,4,3}E_{s,4,2} + 90E_{s,4,3}E_{s,2,2}^{2} + 120E_{s,4,3}E_{s,2,1}^{2}$ $+ 180E_{s,4,2}E_{s,2,1}E_{s,2,2} + 30E_{s,4,4}E_{s,2,2}E_{s,2,1} - 270E_{s,2,2}^{3}E_{s,2,1}$ $- 360E_{s,2,1}^{3}E_{s,2,2}$
	$C_{s,8,4} = E_{s,8,4} - E_{s,4,4}^{2} - 18E_{s,4,2}^{2} - 16E_{s,4,3}^{2} - 54E_{s,2,2}^{4} - 144E_{s,2,1}^{4} - 432E_{s,2,2}^{2}E_{s,2,1}^{2}$ $+ 12E_{s,4,4}E_{s,2,2}^{2} + 96E_{s,4,3}E_{s,2,1}E_{s,2,2} + 144E_{s,4,2}E_{s,2,1}^{2} + 72E_{s,4,2}E_{s,2,2}^{2}$ $+ 96E_{s,4,3}E_{s,2,2}E_{s,2,1}$

Table V-2. Relationships between cumulants and moments [MAR98].

3. Transformations of Moments and Cumulants

The behavior of higher order moments and cumulants to various transformations is an important factor in determining how useful these quantities may be to characterize signals in systems.

a) Translation

The only effect of translation on the received signal is only the mean changes. The variance and all the higher order moments or cumulants remain unaffected.

b) Rotation

The rotation of the received signal's constellation, due to multipath or other distortions, affects the relative variances and higher order moments or cumulants, though certain other parameters such as the eigenvalues and the covariance matrix are invariant to rotation.

VI. INTRODUCTION TO NEURAL NETWORKS

Chapter V discussed the use of higher order statistics as features for digital signal classification. This chapter will give a brief overview of neural networks that will be used to process some of these features in order to identify the various digital modulation sequences. Neural networks are iterative, nonlinear schemes that attempt to imitate the way a human brain works. Rather than using a digital model, in which all computations manipulate zeros and ones, a neural network works by creating connections between basic processing elements called neurons. The organization and weights of the connections determine the output of the neural network.

A. BIOLOGICAL NEURON MODEL

The brain is a collection of about 10 billion interconnected neurons, where each neuron is a cell that uses biochemical reactions to receive, process and transmit information. Figure VI-1 shows a rough drawing of a biological neuron. A neuron's dendritic tree is connected to a thousand neighboring neurons. A positive or negative charge is received by one of the dendrites when one of those neurons fires. The strengths of all the received charges are added together through the processes of spatial and temporal summation. Spatial summation occurs when several weak signals are converted into a single large one, while temporal summation converts a rapid series of weak pulses from one source into one large signal. The aggregate input is then passed to the soma (cell body). The soma and the enclosed nucleus do not play a significant role in the processing of incoming and outgoing data. Their primary function is to perform the continuous maintenance required to keep the neuron functional. The part of the soma that does concern itself with the signal is the axon hillock. If the aggregate input is greater than the axon hillock's threshold value, then the neuron is energized, and an output signal is transmitted down the axon. The strength of the output is constant, regardless of whether the input was just above the threshold, or a hundred times as larger.

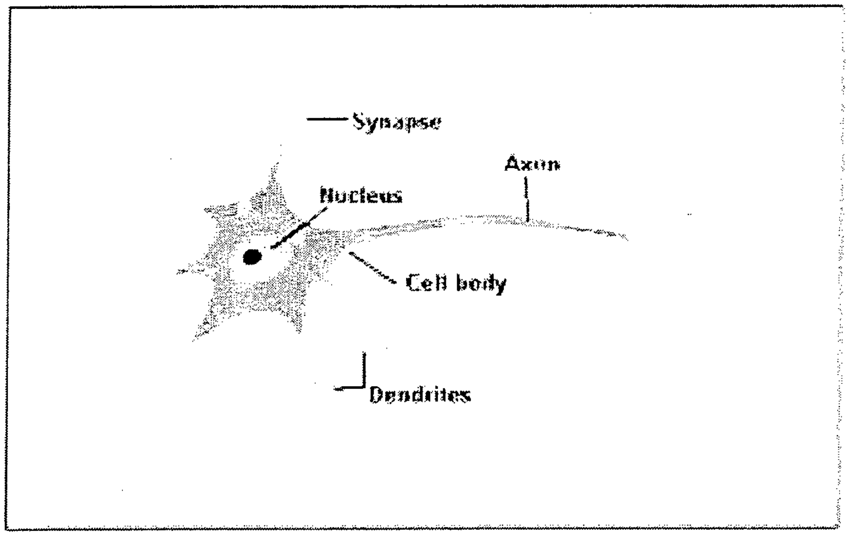


Figure VI-1. Schematic drawing of a biological neuron.

B. ARTIFICIAL NEURON MODEL

As complicated as the biological neuron is, it may be represented by a very simple model, as illustrated in Figure VI-2. The neuron can have any number of inputs p_i which are each multiplied by a weight w_i representing the strength of the contribution to the neuron. Then, all weighted inputs are summed and biased with a value b. This bias is an additional weight associated to a constant input taken equal to one. Bias parameters add additional flexibility to a network by allowing the network hyperplane decision boundary not to be constrained to pass through the origin. Such a constraint usually results in performance degradations, and for this reason neural network implementations most often include bias terms.

In addition, each neuron has a transfer function f that transforms the sum of all weighted inputs to give the final neuron output α. A large variety of linear or nonlinear transfer functions may be selected, and the specific choice depends upon the exact application the neuron is built for. A list of the most common transfer functions is shown in Table VI-1. A neural network usually consist of many interconnected neurons that form serial processing layers, as shown for example in Figure VI-3 which illustrates a feed-forward network. Numerous other configurations exist and further details may be found in [HDB96, Section 19.14].

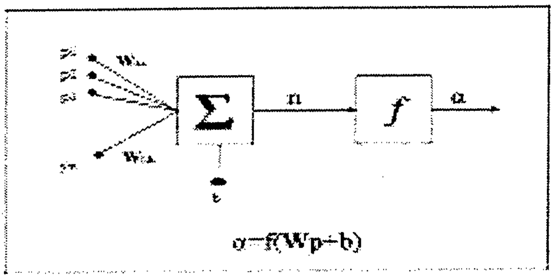


Figure VI-2. Multi Input Neuron Model.

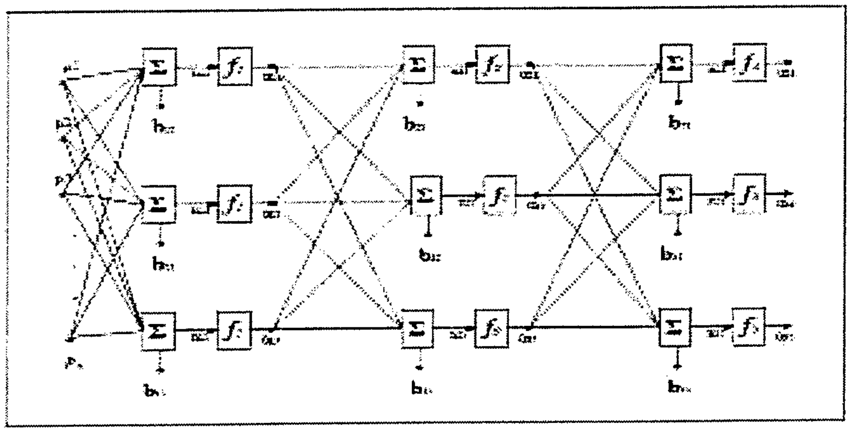
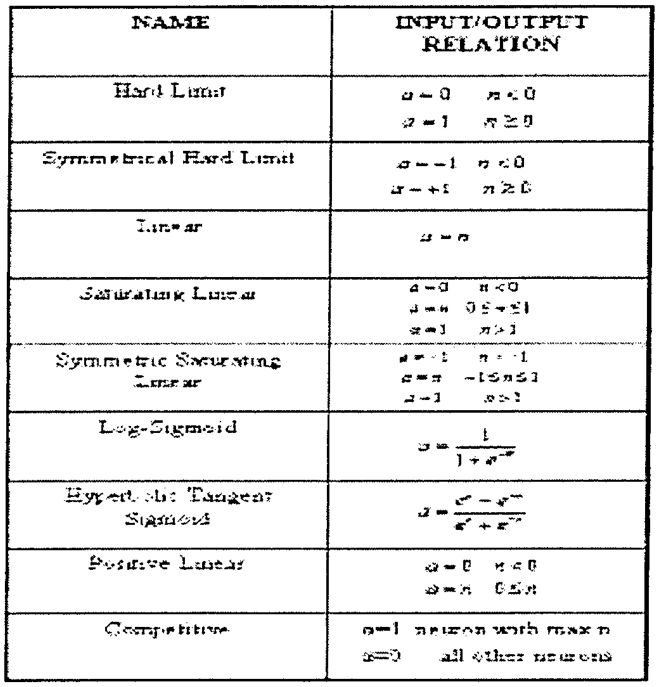


Figure VI-3. Multilayer Neural Network.



NAME	INPUT/OUTPUT RELATION
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$
Linear	$a = n$
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$
Log-Sigmoid	$a = \frac{1}{1 + e^{-n}}$
Hyperbolic Tangent Sigmoid	$a = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}}$
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n$
Competitive	$a=1$ neuron with max n $a=0$ all other neurons

Table VI-1. Activation functions.

C. TYPES OF NEURAL NETWORKS

Many different types of neural networks can be designed to perform a specific task. Some of the more popular types include the multilayer perceptron [HDB96, Section 11-2] which is generally trained with the backpropagation algorithm [HDB96, Section 11-7], learning vector quantization [HDB96, Section 14-16], radial basis function [HDB96, Section 12-2], Hopfield [HDB96, Section 3-12], Kohonen [HDB96, Section 13-15] and others... Some neural networks are classified as feed-forward while others are recurrent (i.e., implement feedback) depending on how data is processed through the network. Another approach to classify neural network types is by learning (or training) method, as some neural networks employ supervised training while others are referred to as unsupervised. In supervised implementation the network is trained using labeled data, i.e., fed with input data with associated known a-priori target outputs. Unsupervised algorithms do not take advantage of labeled data. They essentially perform clustering of the data into similar groups based on the input features characteristics.

VII. DIGITAL MODULATION CLASSIFICATION SCHEME

Chapter VI reviewed the main concepts behind multi-input neural networks. Chapter VII discusses the specific overall classification scheme derived to differentiate between the various digital modulation schemes considered in this work. Note that we take into account effects due to additive Gaussian noise and multi-path environment. Our classification scheme combines a hierarchical approach, where one or two specific features are used to separate between given sets of classes. Such separation is done with simple neural networks which are used to "model" the effects due to additive Gaussian noise and multi-path in the transmission channel.

The features selected to differentiate between the various digital modulation schemes considered in our work are a combination of moments and cumulants. We discussed in Chapter V the concepts of higher-order moments and cumulants, and reviewed earlier work proposed by Marchand who investigated a cumulant-based modulation classification [MML97]. Specifically, Marchand calculated theoretical values for moments and cumulants up to the 8th order for 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM and 256-QAM schemes [MAR98, p. 178, Table B.1]. These values have been verified and corrected for minor sign errors and are presented in Tables VII-1 through VII-8. Note that all moments and cumulant values are normalized by the theoretical signal power P.

		2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2nd order	$\frac{E_{S,2,2}}{P}$	1	0	0	0	0	0
moments	$\frac{E_{S,2,1}}{P}$	1	1	1	1	1	1

Table VII-1. Theoretical 2nd order moment values for 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM and 256-QAM modulations.

		2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
4th order	$\frac{E_{S,4,4}}{P^2}$	1	1	0	-0.68	-0.619	-0.604
moments	$\frac{E_{S,4,3}}{P^2}$	1	0	0	0	0	0
	$\frac{E_{S,4,2}}{P^2}$	1	1	1	1.32	1.38	1.395

Table VII-2. Theoretical 4th order moment values for 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM and 256-QAM modulations.

		2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
6th order	$\frac{E_{S,6,6}}{P^3}$	1	0	0	0	0	0
moments	$\frac{E_{S,6,5}}{P^3}$	1	1	0	-1.32	-1.298	-1.288
	$\frac{E_{S,6,4}}{P^3}$	1	0	0	0	0	0
	$\frac{E_{S,6,3}}{P^3}$	1	1	1	1.96	2.22	2.29

Table VII-3. Theoretical 6th order moment values for 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM and 256-QAM modulations.

		2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
8th order	$\frac{E_{S,8,8}}{P^4}$	1	1	1	2.2	1.91	1.82
moments	$\frac{E_{S,8,7}}{P^4}$	1	0	0	0	0	0
	$\frac{E_{S,8,6}}{P^4}$	1	1	0	-2.48	-2.75	-2.81
	$\frac{E_{S,8,5}}{P^4}$	1	0	0	0	0	0
	$\frac{E_{S,8,4}}{P^4}$	1	1	1	3.12	3.96	4.19

Table VII-4. Theoretical 8th order moment values for 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM and 256-QAM modulations.

		2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2nd order	$\frac{C_{S,2,2}}{P}$	1	0	0	0	0	0
cumulants	$\frac{C_{S,2,1}}{P}$	1	1	1	1	1	1

Table VII-5. Theoretical 2nd order cumulant values for 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM and 256-QAM modulations.

		2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
4th order	$\frac{C_{S,4,4}}{P^2}$	-2	-1	0	-0.68	-0.619	-0.604
cumulants	$\frac{C_{S,4,3}}{P^2}$	-2	0	0	0	0	0
	$\frac{C_{S,4,2}}{P^2}$	-2	-1	-1	-0.68	-0.619	-0.604

Table VII-6 Theoretical 4th order cumulant values for 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM and 256-QAM modulations.

		2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
6th order	$\frac{C_{S,6,6}}{P^3}$	16	0	0	0	0	0
cumulants	$\frac{C_{S,6,5}}{P^3}$	16	-4	0	2.08	1.797	1.734
	$\frac{C_{S,6,4}}{P^3}$	16	0	0	0	0	0
	$\frac{C_{S,6,3}}{P^3}$	16	4	4	2.08	1.797	1.734

Table VII-7. Theoretical 6th order cumulant values for 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM and 256-QAM modulations.

		2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
8th order	$\frac{C_{S,8,8}}{P^4}$	-244	-34	1	-13.98	-11.5	-10.97
cumulants	$\frac{C_{S,8,7}}{P^4}$	-244	0	0	0	0	0
	$\frac{C_{S,8,6}}{P^4}$	-244	0	0	-29.82	-27.078	-26.438
	$\frac{C_{S,8,5}}{P^4}$	-244	0	0	0	0	0
	$\frac{C_{S,8,4}}{P^4}$	-244	-17	-17	17.379	24.11	25.704

Table VII-8. Theoretical 8th order cumulant values for 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM and 256-QAM modulations.

A. FEATURE EXTRACTION

A closer look to Tables VII-1 through VII-8 reveals that some moments and cumulants can be used to separate different modulation schemes while others have little or no use. For example, the 6th order moment $E_{s,6,5}$ can theoretically be used to differentiate the 8-PSK scheme from all others.

Note that at this point it is essential to remember that Tables VII-1 to VII-8 present the theoretical values obtained for moment and cumulants, i.e., obtained assuming the signal is clean and of infinite length. However, in practice signals are usually subject to some type of distortion, either inside the transmitter or during transmission, and are of finite length. In addition, channel distortion is likely to affect the higher order statistics of the signal, although moments and cumulants are relatively robust to signal distortion [Chapter V, Section B, Paragraphs 3.a and 3.b]. Moreover, no infinite dataset is available in practical applications, and finite data length can significantly affect the estimates accuracy.

1. Signal Sequences Creation

Each signal used in this study was generated using MATLAB. We assumed that carrier frequencies were estimated correctly and the signals heterodyned down. Thus, we only considered complex baseband signals. The modulation types considered in this work include 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM, 64-QAM and 256-QAM, previously considered by Marchand [MAR98], and 2-FSK, 4-FSK and 8-FSK. A total of 100,000 samples per modulation scheme were created and stored. A typical bit rate of 1Mbps was chosen for all simulations. The sampling frequency was chosen in such a way that all schemes are sampled with 4 samples/symbol, a number currently used by manufacturers of modulation and demodulation devices [COP00]. The digital information (message) is generated randomly for every trial, to ensure results are independent of the message transmitted.

2. Moments and Cumulants Estimation

Estimating moment and cumulant values for all modulation schemes considered is based on the theoretical formulas provided in Tables V-1 and V-2. For this process, only the moments and cumulants that show some special characteristics as class features are selected. The estimation is done on a subset of 20,000 samples per scheme, out of the total 100,000 samples per scheme dataset. Two different cases are examined. First, the signals are generated noise-free. Second, the signals are distorded by additive white Gaussian noise (AWGN) to form a SNR equal to 0 dB. Estimated cumulants and moments are presented in Table VII-9, where the values shown in parenthesis are those corresponding to the 0 dB case.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
$\frac{E_{S,2,2}}{P}$	0.5 (0.24)	0.25 (0.12)	0.25 (0.12)	1 (0.5)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
$\frac{E_{S,4,4}}{P^2}$	1 (0.23)	0.5 (0.12)	0.25 (0.06)	1 (0.25)	1 (0.25)	0 (0)	-0.68 (-0.16)	-0.61 (-0.16)	-0.6 (-0.51)
$\frac{E_{S,4,3}}{P^2}$	0.5 (0.5)	0.25 (0.25)	0.25 (0.25)	1 (1)	0 (0)	0 (0)	0 (0)	0.01 (0)	0.002 (0.0004)
$\frac{E_{S,4,2}}{P^2}$	1 (1.75)	1 (1.75)	1 (1.75)	1 (1.75)	1 (1.75)	1 (1.75)	1.32 (1.82)	1.38 (1.85)	1.34 (1.85)
$\frac{E_{S,6,5}}{P^3}$	1 (0.75)	0.5 (0.35)	0.25 (0.18)	1 (0.75)	1 (0.72)	0 (0)	-1.32 (-0.6)	-1.29 (-0.6)	-1.28 (-0.54)
$\frac{E_{S,8,8}}{P^4}$	1 (0.25)	1 (0.5)	0.5 (0.18)	1 (0.13)	1 (0.18)	1 (0.07)	2.2 (0.08)	1.91 (0.11)	1.82 (0)
$\frac{E_{S,8,6}}{P^4}$	1 (2.57)	0.5 (1.18)	0.25 (0.7)	1 (2.64)	1 (2.61)	0 (0.1)	-2.48 (-2.37)	-2.75 (-2.5)	-2.81 (-2.25)
$\frac{E_{S,8,4}}{P^4}$	1 (12.82)	1 (12.91)	1 (13.13)	1 (13.02)	1 (13)	1 (13)	3.12 (15.5)	3.96 (15.9)	4.19 (16.18)
$\frac{C_{S,4,4}}{P^2}$	-0.1 (0)	-0.5 (-0.11)	-0.5 (-0.13)	-2 (-0.5)	-1 (-0.25)	0 (-0.25)	-0.68 (-0.17)	-0.619 (-0.15)	-0.604 (-0.15)
$\frac{C_{S,6,5}}{P^3}$	1 (0)	-0.8 (0)	0.25 (0)	16 (2)	-4 (-0.5)	0 (0)	2.08 (0.25)	1.797 (0.23)	1.734 (0.22)
$\frac{C_{S,8,8}}{P^4}$	31.6 (2.22)	2.45 (0.52)	2.45 (0.1)	-244 (-15.5)	-34 (-1.84)	1 (0)	-13.98 (0)	-11.5 (-0.86)	-10.97 (-0.8)
$\frac{C_{S,8,4}}{P^4}$	-64.5 (66)	-28 (65.82)	-28.7 (66.6)	-244 (13.02)	-18 (65.5)	-17 (65.73)	17.37 (15.5)	24.11 (76.02)	24.7 (76.18)

Table VII-9. Estimated values for selected moments and cumulants up to the 8th order for 2-FSK, 4-FSK, 8-FSK, 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM and 256-QAM modulation schemes; total samples per scheme=20,000. SNR=∞, SNR =0 dB shown in parentheses, P= noisy signal power.

Table VII-9 shows that there are small differences between the theoretical and the estimated values of moments and cumulants for the noise-free case but in general the values are quite close. However, this is no longer true for the 0 dB case. Such changes are mainly due to the noise impact in the estimated noisy signal power, and to a smaller extent to the noise effects in the moment and cumulant estimation process. For example, note that $C_{s,8,8}$ exhibits a large deviation from its noise-free value, making the selection of the most appropriate feature even more difficult.

3. Feature Selection

Based on the results of Table VII-9, an initial selection of the features with the most interesting characteristics is made. These features are presented in Table VII-10 below.

$\frac{C_{S,8,8}}{P^4}$	$\frac{E_{S,2,2}}{P}$	$\frac{E_{S,4,3}}{P^2}$	$\frac{E_{S,6,5}}{P^3}$	$\frac{C_{S,6,5}}{P^3}$
Separates 2-PSK from all other schemes	**Separates M-QAM 4-PSK and 8-PSK from M-FSK**	**Separates M-QAM 4-PSK and 8-PSK from M-FSK**	**Separates 4-FSK from 8-FSK**	**Separates 4-PSK and 8-PSK, from M-QAM**
				Separates 2-FSK from 4-FSK and 8-FSK

Table VII-10. Selection of the most discriminating features for the proposed scheme classification.

Further testing of the robustness of those features is essential to determine their usefulness in a classification scheme and should include SNR level variations and distortions due to fading and multipath.

a) Robustness to White Noise

We first investigated the robustness of features to additive white Gaussian noise, i.e., the AWGN propagation model case as described earlier in Chapter III, section B, paragraph (a). We considered all modulation types in SNR levels between 0 and 20dB with 100 trials per SNR level, and various data length for cumulant and moment estimation. Complete results are presented in Appendix E. Figures VII-1 through VII-5 present the behavior for all selected features as a function of the SNR level for a 15,000 samples dataset.

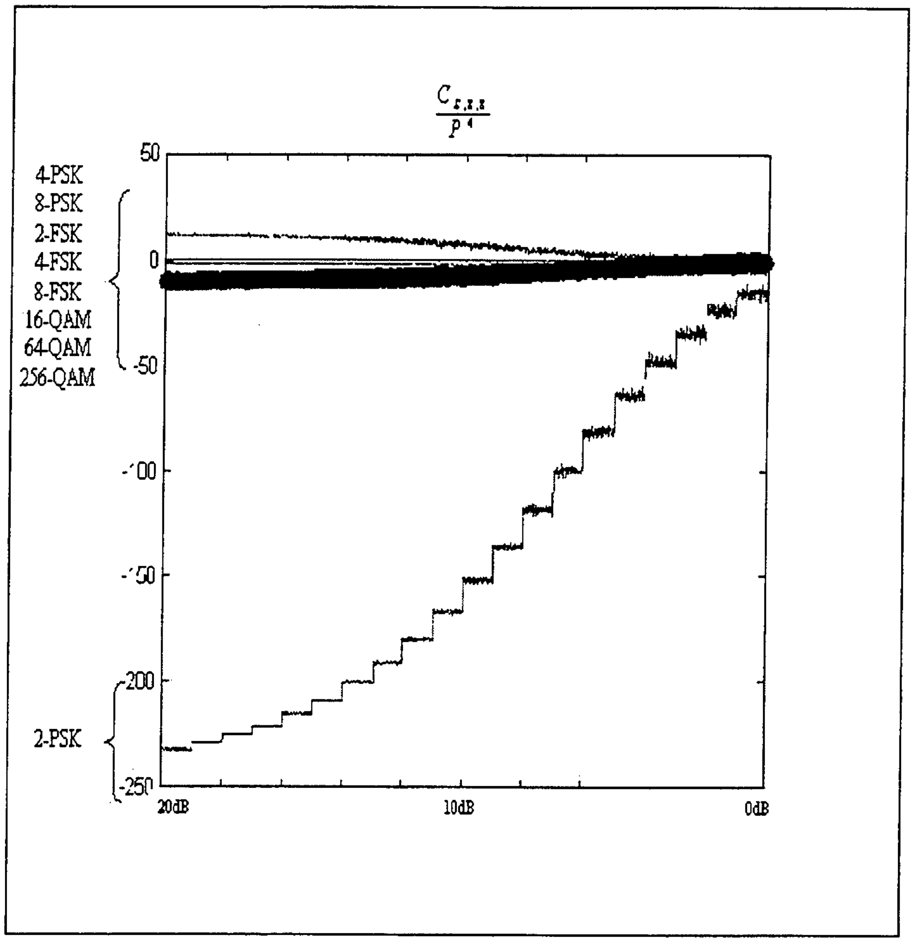


Figure VII-1. $\frac{C_{S,8,8}}{P^4}$ for all modulation schemes; 15,000 samples dataset, 100 trials per SNR level.

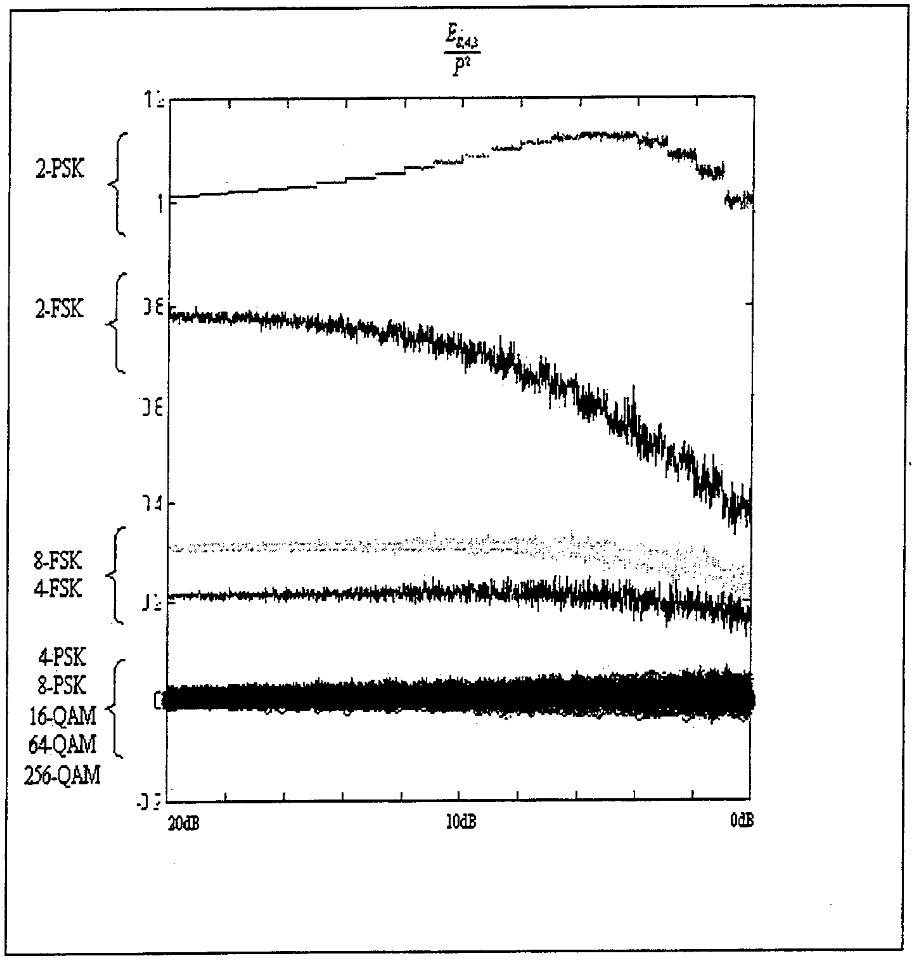


Figure VII-2. $\frac{E_{S,4,3}}{P^2}$ for all modulation schemes; 15,000 samples dataset, 100 trials per SNR level.

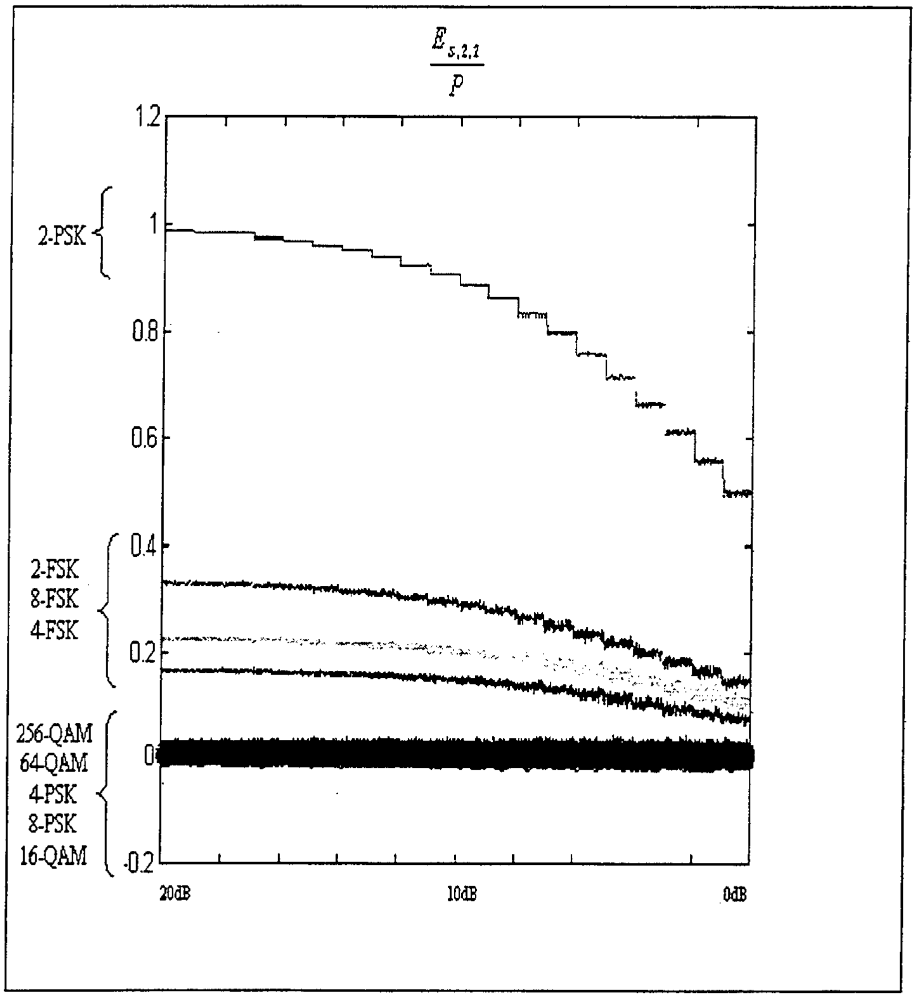


Figure VII-3. $\frac{E_{s,2,2}}{P}$ for all modulation schemes; 15,000 samples dataset, 100 trials per SNR level.

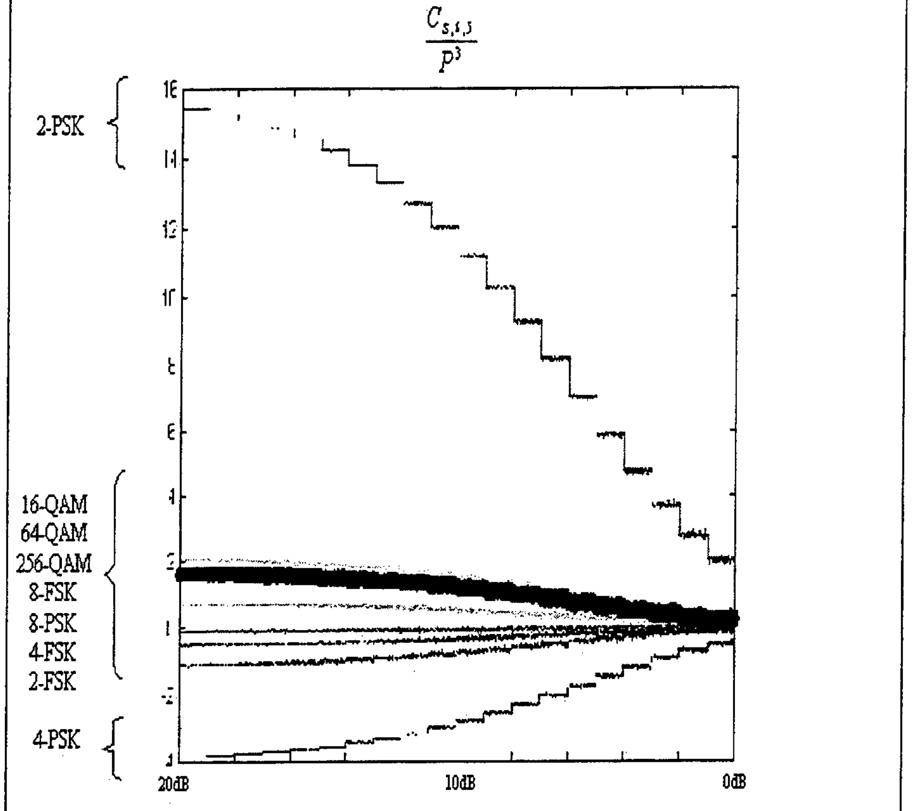


Figure VII-4. $\frac{C_{S,6,5}}{P^3}$ for all modulation schemes; 15,000 samples dataset, 100 trials per SNR level.

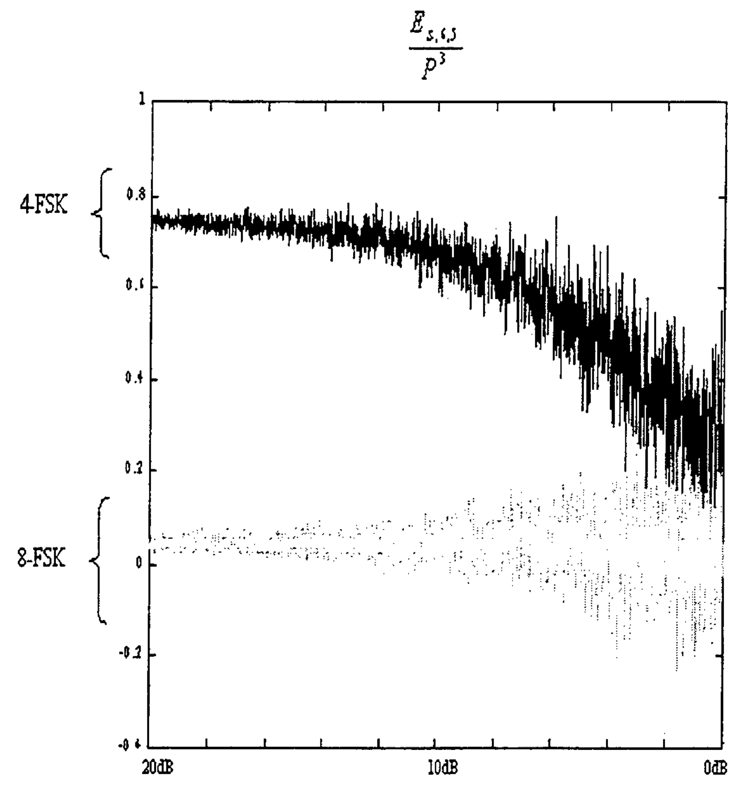


Figure VII-5. $\frac{E_{S,6,5}}{P^3}$ for all modulation schemes; 15,000 samples dataset, 100 trials per SNR level.

Figures VII-1 through VII-5 show the selected features may be used to separate all schemes, except M-QAM, down to almost 5dB and in some cases even lower. However the AWGN channel is a simplified case that does not take into account fading and multipath propagation phenomena.

b) Robustness to Fading and Multi-path Environments

Robustness of the selected features was investigated next by studying their behavior when the modulation signal is passed through the various fading and multipath propagation models covered in Chapter III, Sect. B.1-4. The specific impulse responses for each propagation channel used in this study are presented in Appendix C (Channels 1 to 9). These channels cover a variety of different environments, from rural environment models with 1 or 2 paths to urban models with more than 3 different propagation paths. SNR levels between 0 to 20dB were again considered here, and 100 trials implemented per SNR. Complete results are presented in Appendix E. Figures VII-6 through VII-10 present the behavior for all selected features as a function of the SNR level for a 15,000 samples dataset.

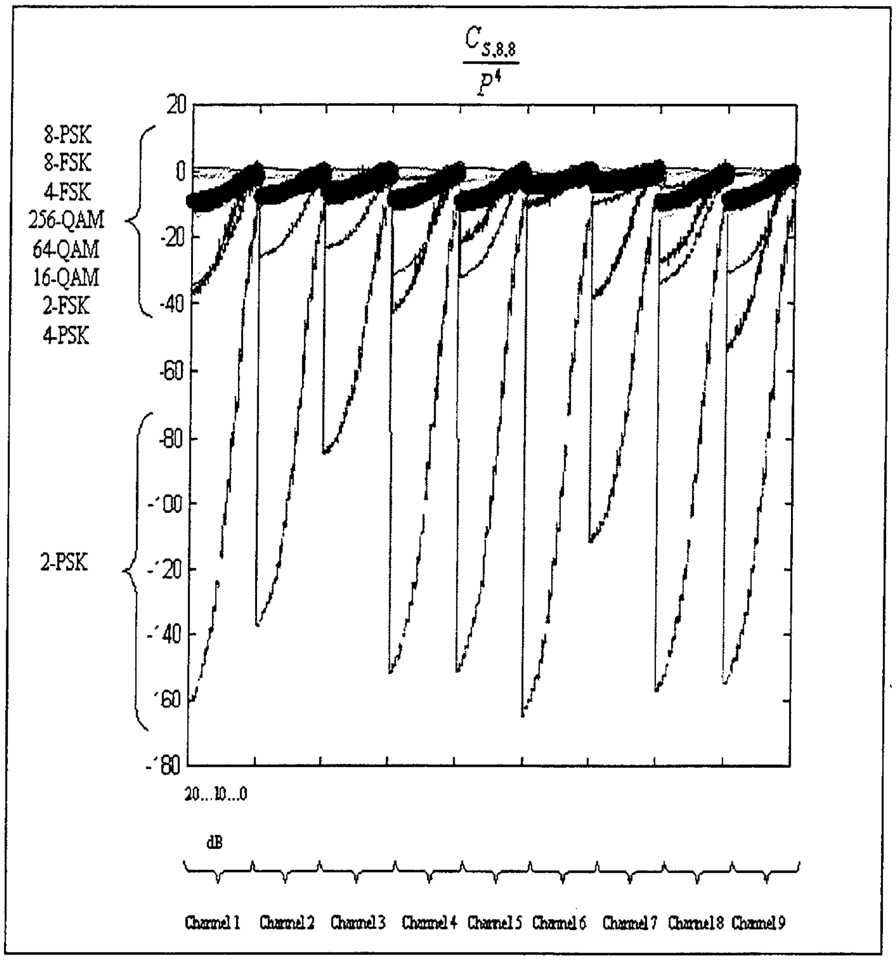


Figure VII-6. $\frac{C_{S,8,8}}{P^4}$ for all modulation schemes; 15,000 samples dataset, 100 trials per SNR level.

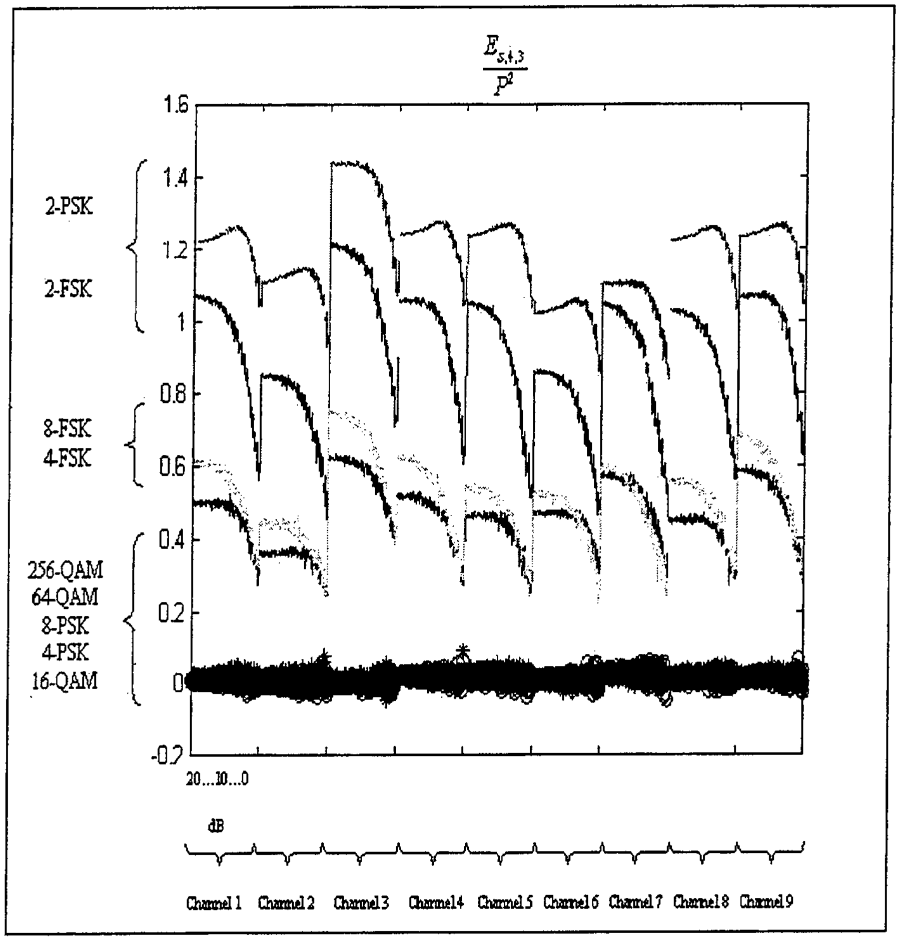


Figure VII-7. $\frac{E_{S,4,3}}{P^2}$ for all modulation schemes; 15,000 samples dataset, 100 trials per SNR level.

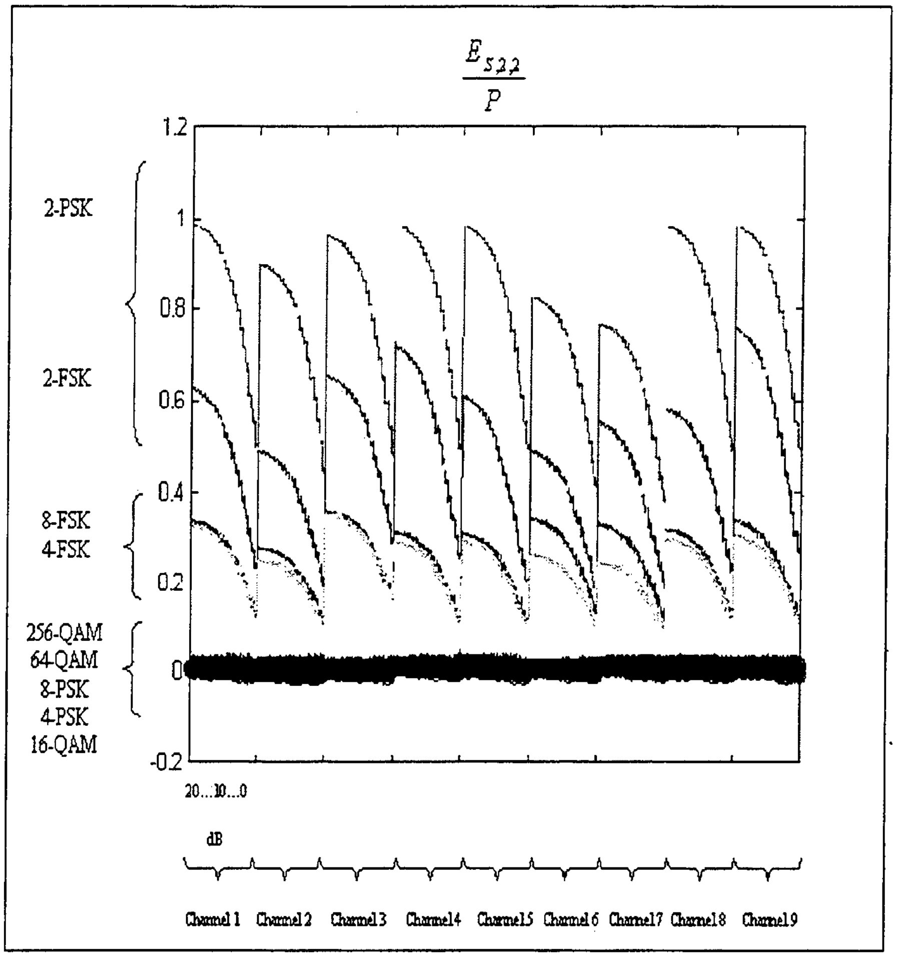


Figure VII-8. $\frac{E_{S,2,2}}{P}$ for all modulation schemes; 15,000 samples dataset, 100 trials per SNR level.

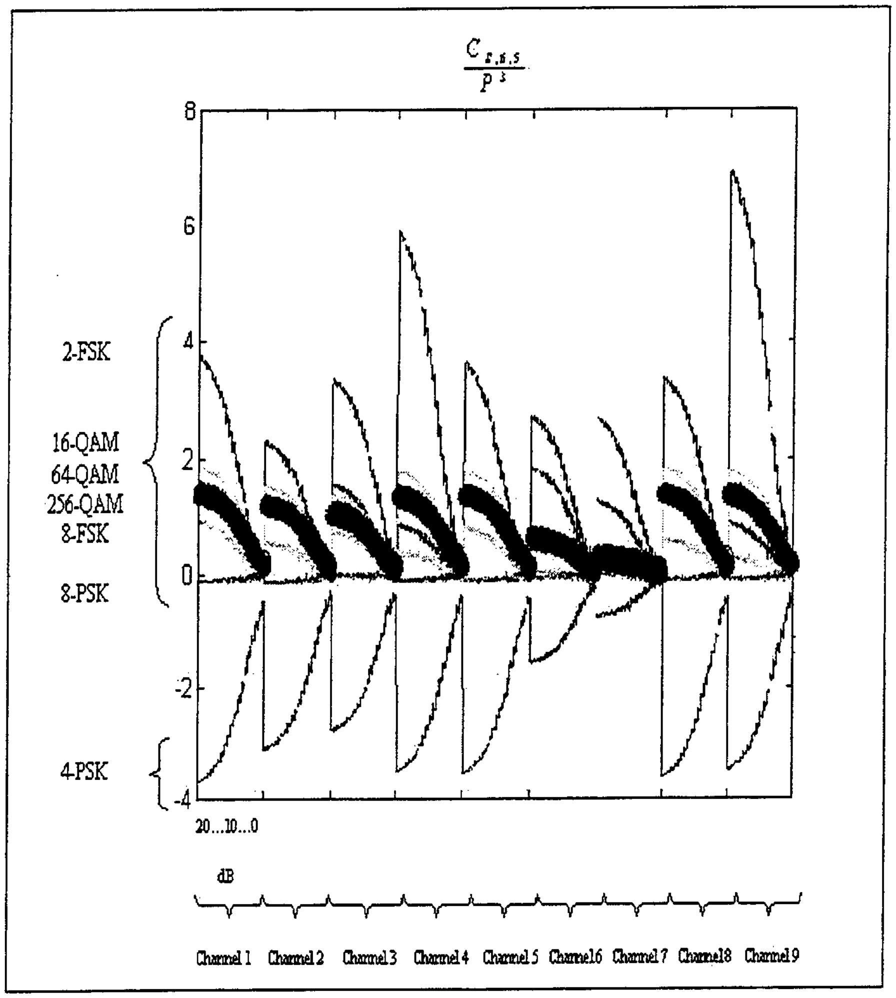


Figure VII-9. $\frac{C_{S,6,5}}{P^3}$ for all modulation schemes; 15,000 samples dataset, 100 trials per SNR level.

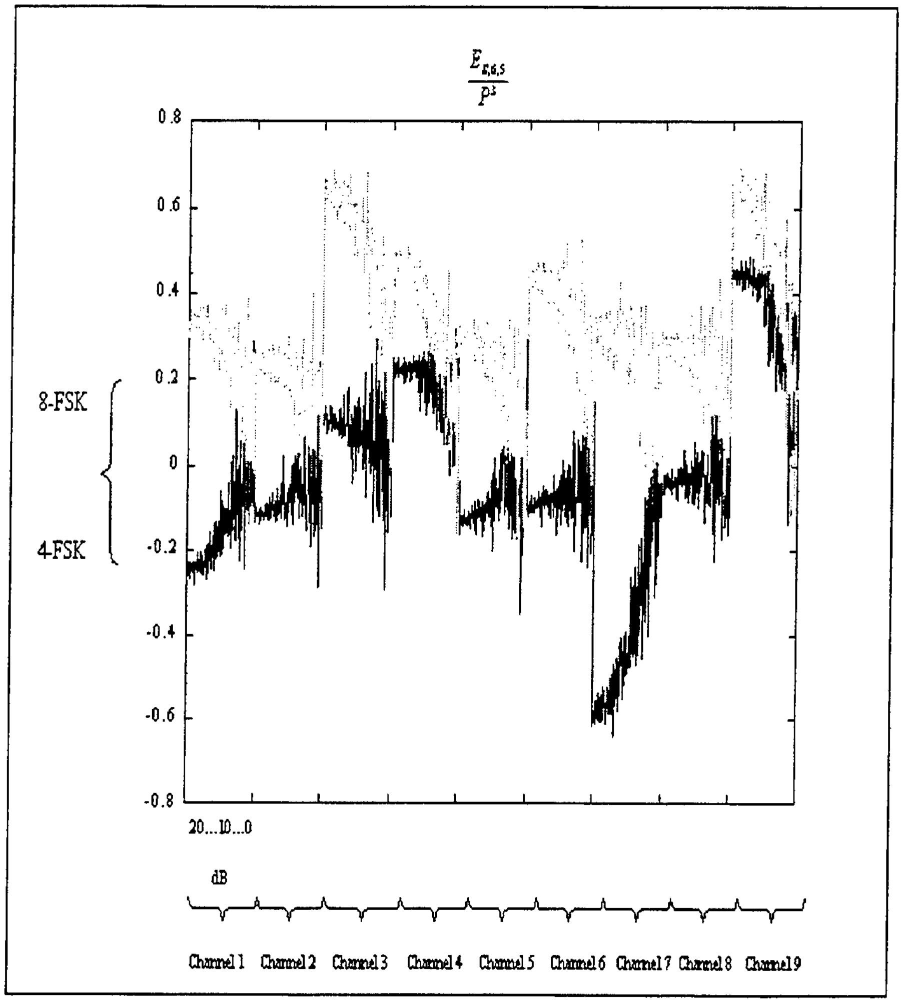


Figure VII-10. $\frac{E_{S,6,5}}{P^3}$ for all modulation schemes; 15,000 samples dataset, 100 trials per SNR level.

Figures VII-6 through VII-10 reveal the impact of the modeled wireless propagation channel on the higher order statistics of the modulation types. Some propagation channels (Channels 3 and 7) distort the selected features to such an extent that any attempt to built a classification scheme based on fixed class feature thresholds is doomed to fail.

B. PROPOSED SCHEME

Figures VII-1 through VII-10 show that the proposed classification scheme has to be flexible to SNR level and propagation channel distortions. With the exception of the M-QAM modulations, higher-order statistics may have the power to separate different modulations provided one introduces some type of "agile" classification scheme. At this point, neural networks seemed a logical approach to the problem because they offer flexibility and performance proportional to the quality of the training data set available. In addition, neural networks can be a very fast, near real-time, solution to the problem, once they are trained. Note that the classification of M-QAM type is still a problem since no suitable higher-order statistics can be found to serve as classification features, for the varying environments considered. In this case, a combination of blind equalization techniques, previously considered by Barbarossa et. al. [BAR00] will be used to identify the specific M-QAM type. The proposed method cascades the FSE-CMA equalization and the AMA method, previously described in Chapter IV. The complete classification scheme is shown in Figure VII-11.

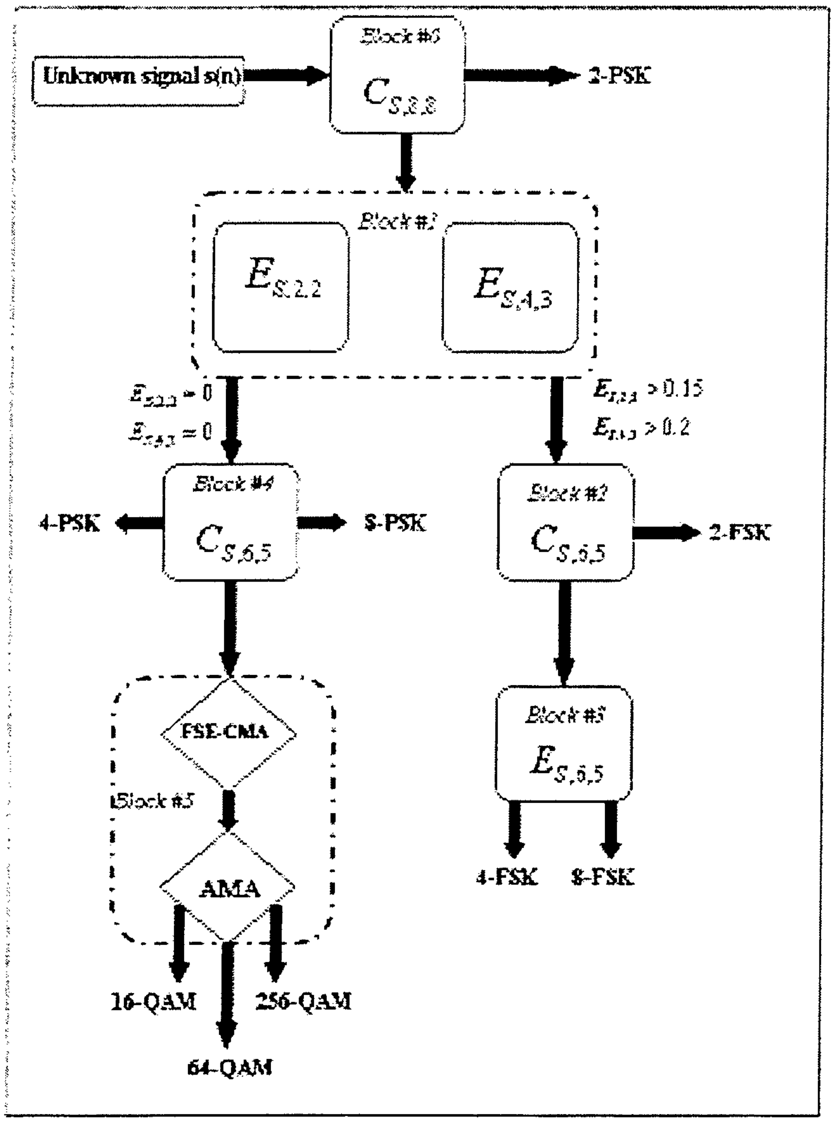


Figure VII-11. Theoretical classification scheme for 2-FSK, 4-FSK, 8-FSK, 2-PSK, 4-PSK, 8-PSK, 16-QAM, 64-QAM & 256-QAM modulation types.

The overall classification scheme consists of five high-order statistic based classification blocks that are described next, and one equalization based block. The first five blocks contain basic back-propagation neural network classifiers trained to identify all constant modulus signal types: 2-FSK, 4-FSK, 8-FSK, 2-PSK, 4-PSK, 8-PSK, and generic M-QAM types. The specific identification of the QAM type (16-QAM, 64-QAM, 256-QAM) is accomplished via a combination of FSE-CMA and AMA equalization methods. Note that the use of the FSE-CMA is essential for the proper initialization of the AMA algorithm [BSC98].

1. Neural Network Blocks Implementation

Conceptually, the proposed classification scheme includes two different approaches. The neural network classifiers and the blind equalization classifier.

Blocks number 0 to 4 in Figure VII-11 are single, two, three or four layer neural networks. Each network is trained with a specific feature training sequence, with the exception of the second block that is trained with two features simultaneously. The number of layers, the activation functions and the number of epochs vary from block to block. The choice for the specific characteristics of each network was done empirically by trial and error and based on the clarity of the specific feature. Note that more layers and more epochs were selected for features more severely distorted from noise or propagation channel effects than others. Table VII-11 presents the characteristics for each neural network.

	BLOCKS				
	#0	*#1*	*#2*	*#3*	*#4*
Inputs	1	2	1	1	1
Classifying Feature(s)	$\frac{C_{S,8,8}}{P^4}$	$\frac{E_{S,4,3}}{P^2}, \frac{E_{S,2,2}}{P}$	$\frac{C_{S,6,5}}{P^3}$	$\frac{E_{S,6,5}}{P^3}$	$\frac{C_{S,6,5}}{P^3}$
Layers	2	3	3	4	3
Arrangement of neurons per layer	8-1	20-8-1	20-10-1	14-4-2-1	20-10-1
Activation function per layer	'tansig' 'satlins'	'tansig' 'tansig' 'purelin	'tansig' 'tansig' 'satlins'	'tansig' 'tansig' 'tansig' 'purelin'	'tansig' 'tansig' 'satlins'
Required epochs	40	40	70	100	40

Table VII-11. Neural network characteristics for blocks #0 through #4.

Training data was generated according to the schematic shown in Figure VII-12. First, a 15,000 samples sequence was extracted out of the 100,000 samples generated for each modulation type, as described in Chapter VII, Sect. A.1. Next, each sequence was passed through one out of nine different propagation channels further described in Appendix C (channels 1 to 9). These channels were selected to represent a wide variety of propagation situations. They include from single to more than 4-path models that correspond to rural, small town or urban propagation conditions. Next, the resulting signal sequences were corrupted with additive white Gaussian noise with SNR levels between 0 to 20dB. Finally, 100 trials per SNR level were generated. Note that we used multiple trials per SNR level to get a sense of the variance in the measurements and enhance the network's performance.

Next, the selected features defined above were estimated for each noisy signal. As a result, each dataset was associated with six different feature parameters and each feature (or combination of) fed into the appropriate network for training. Figure VII-12 shows the training dataset creation process.

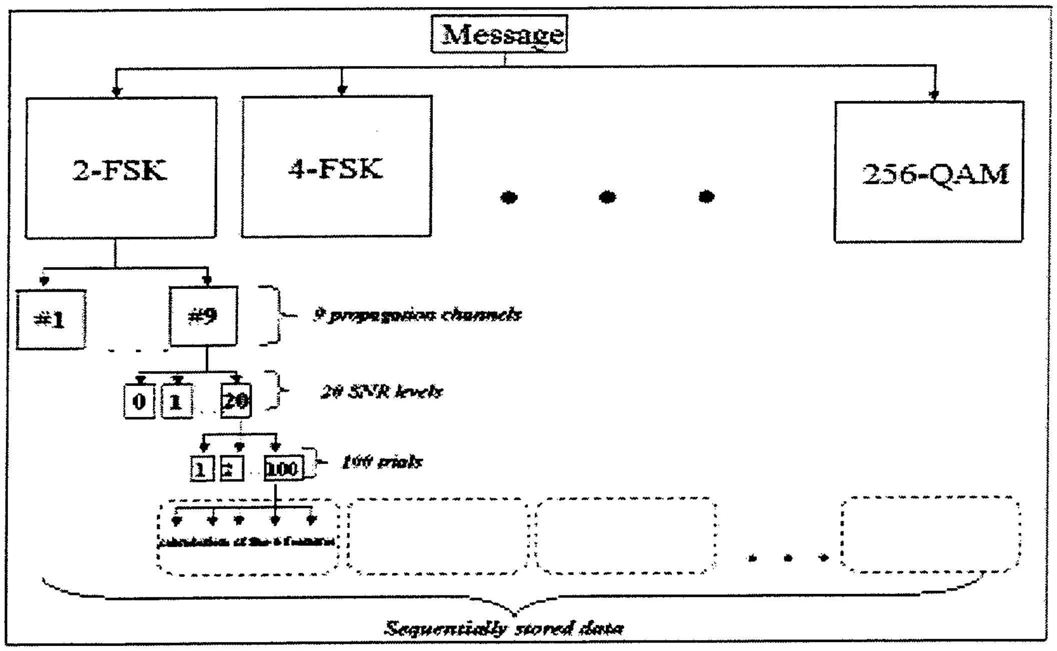


Figure VII-12. Training schematic for the neural network based classification blocks of the overall classification scheme.

2. FSE-CMA & AMA Classifier Block Set Implementation

The purpose of the last block (Block #5) is to differentiate within the QAM family, where 16-QAM, 64-QAM and 256-QAM signal types are considered here. These modulation types are those most susceptible to noise and fading due to the proximity of the associated constellation's centroids, especially for higher order constellations. Recall that Table VII-11 showed how similar the higher-order parameters are for QAM schemes, thereby making them of little use in classification applications.

Block #5 consists of two parts. The incoming M-QAM signal is first equalized using the FSE-CMA algorithm, as described in Chapter IV, Sect. A. This method is proved to be efficient when the equalized constellation is unknown. A 20-tap equalizer is chosen and the step size selected to be equal to 0.5 to insure the algorithm is stable.

The second process in Block #5 is the AMA algorithm described in Chapter IV, Sect. B. Following the model of Figure IV-9, three different equalizers banks are created, each one matched to one of the three QAM constellations. The parallel model is adopted as it speeds up the decision process, although a model with three AMA equalizers in series would also work. The processed signal obtained after the FSE-CMA step is processed so that all the signal's values lie between –1 and 1 and then passed through the three AMA equalizer banks. Each AMA equalizer is matched to a specific QAM type: 16-QAM, 64-QAM, or 256-QAM. The cost function J(n) given in Equation 4.7 is evaluated after converge for each AMA equalizer. Recall that the theoretical cost function will be smallest when assigned to the correct constellation type, as described in

Chapter IV, Sect. B1. As a result, the constellation type decision is made by picking that which leads to the smallest estimated cost function out of the three computed.

C. TESTING PROCESS

1. Non Linear Case

The proposed classification scheme is ready for testing once all neural networks are trained. The main program, *MAIN_MENU.m*, allows the user to either perform a testing simulation manually, by selecting the unknown signal, SNR and propagation channel, or automate the entire process by considering all modulation types, seven SNR levels ranging between 2dB and 20dB, 50 independent trials for each case and 3 out of the 6 available testing propagation channels. For every trial, a new random message and noise is created to ensure the independence of all results. The 3 propagation channels that are chosen for testing are channels 10, 12 and 14 (Figures C-10, C-12 and C-14) and represent a rural, a small town and urban propagation environments respectively.

The automated process creates seven confusion matrices per propagation channel (one per SNR level), which are presented in Appendix F. These simulations cover a wide spectrum of possible noise and propagation environment combinations. The quantities of interest were the overall classifier performance and the performance of the neural-network (NN)-only portion of the classification set-up, which only considers the generic QAM family but does not subdivide into the three QAM schemes considered here. Figures VII-13 to VII-15 show these two quantities for the classification set-up obtained for a rural area propagation model (Figure C-10), a small town propagation model (Figure C-12) and an urban propagation model (Figure C-15). Results show the NN-only portion of the classifier performs very well down to 11dB for all cases. At the same time the performance of the complete classifier is steadily lower, especially in low signal to noise ratios, revealing the difficulties of M-QAM separation at low SNR levels.

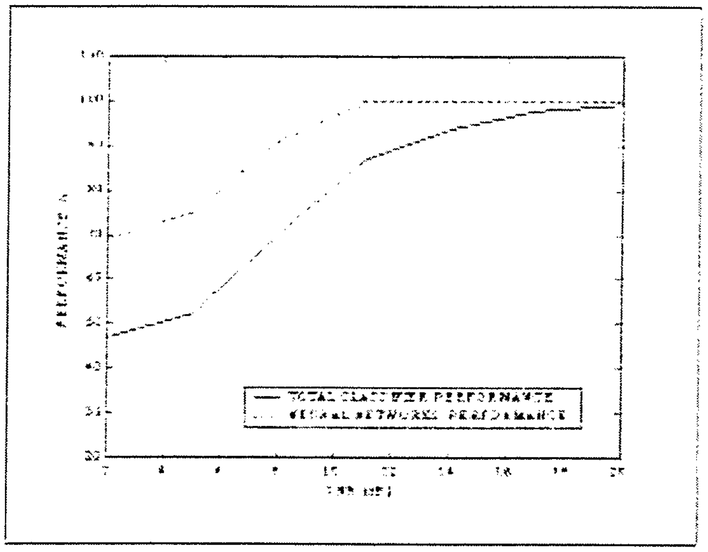


Figure VII-13. Classification performances for channel 10 (Figure C-10); 50 trials per signal per SNR level.

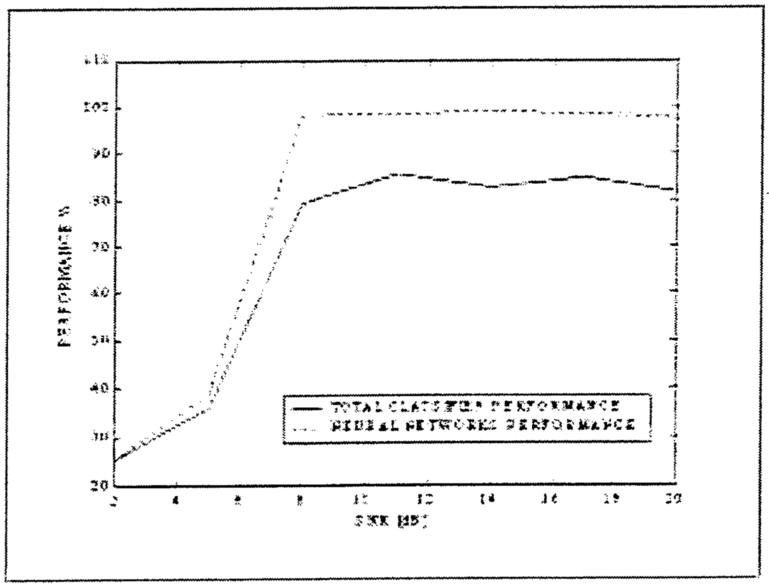


Figure VII-14. Classification performances for channel 12 (Figure C-12); 50 trials per signal per SNR level.

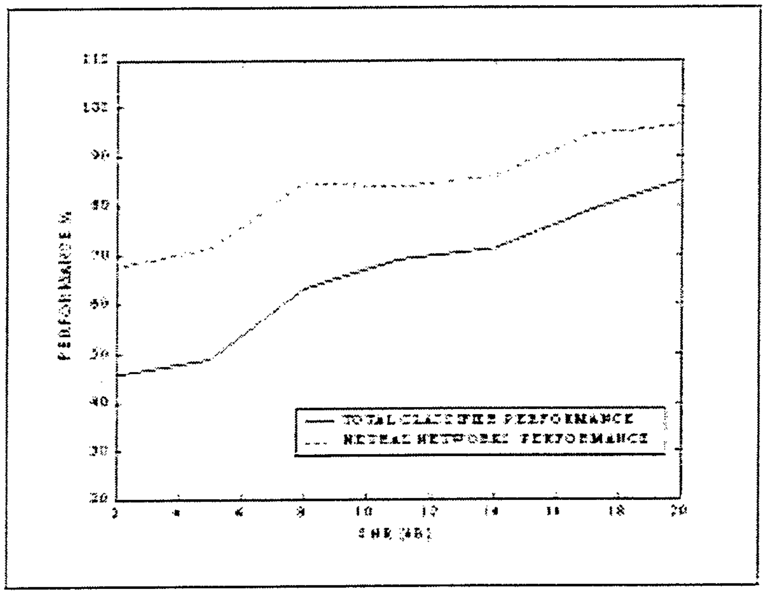


Figure VII-15. Classification performances for channel 15 (Figure C-15); 50 trials per signal per SNR level.

As seen from Figures VII-13 to VII-15, the performances are satisfying for all tested environments down to 11dB approximately. Even in the urban channel model (Figure C-15) the classifier performs well. This is not surprising since the training of all neural network blocks included urban propagation channels (Figures C-3, C-4 and C-7). Simulations also showed that block #5 (designed to separate between the various MQAM schemes) has a consistently lower performance than the rest of the classifier. Such a degradation is due to the fact that the equalization algorithms cannot completely undo non-linear channel effects and mitigate the noise effects. As a result, next we considered a linear channel case to investigate the sensitivity of the equalization steps to a "better behaved" transmission scenario.

2. Linear Case

To investigate the robustness of block #5 to channel distortions, we consider a simple linear channel with impulse response h=[0.9,0.1,0.4] to train the previous network in SNR levels between 2 and 20dB. Next, the network is tested for data transmitted through another linear channel with impulse response c=[1,0,0.5]. As before, 100 trials per SNR level are selected for training, while 50 trials are generated for testing, resulting in seven confusion matrixes (one for each SNR level). Average classification performances are shown in Figure VII-16 and the confusion matrixes included in Appendix F.

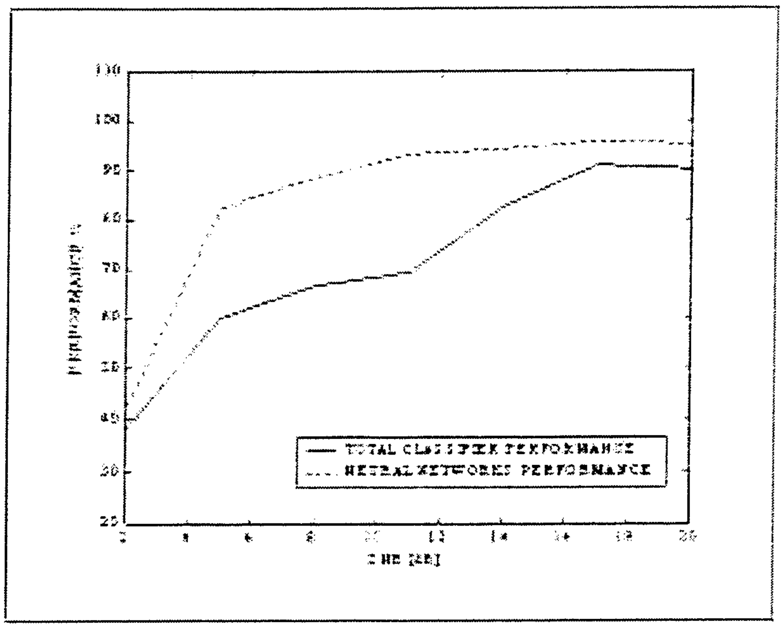


Figure VII-16. Classification performances for network trained on linear channel c=[1,0,0.5]; 50 trials per signal per SNR level.

Figure VII-16 illustrates the fact that the equalization-based classification portion performs better in medium to high SNR levels when channel distortions are linear, as expected.

VIII. CONCLUSIONS

Classifying modulation types has been studied extensively over the last decade as applications arise in numerous different areas. However, few published works deal with real-world propagation models. This study considered the classification of various M-PSK, M-FSK, and M-QAM modulation types under unfavorable propagation conditions and additive white Gaussian noise distortions. We first reviewed the literature in the general area of modulation classification. Initial work indicated that higher-order statistic parameters could be selected to differentiate between all digital modulation types considered in this study when dealing with ideal transmission conditions. However, initial work also showed that these class features were no longer useful in differentiating between specific QAM types when the signals had been distorted by multipath environments.

As a result, a hierarchical classification scheme based on neural network decision nodes was adopted to separate all modulation types, except specific M-QAM types. Classification of various M-QAM types was obtained by a combination of two equalization schemes: the CMA-FSE and the AMA algorithms. While the CMA-FSE is a blind equalization scheme, the AMA takes advantage of the specific M-QAM constellation structure of the QAM types considered. Such a two-step process was motivated by the high sensitivity of QAM modulation types to channel distortions, and the inability of higher-order statistics to separate within the M-QAM family for medium and low SNR levels.

We investigated classification performances for the overall classification scheme in various types of propagation channels (rural, small town and urban) and SNR levels. Results show the classifier performs well for all modulation types considered, but break down fast as the SNR level goes down for M-QAM modulation types. However, such a result was expected as M-QAM modulation types, especially those of high order, are extremely sensitive to noise and multipath fading situations.

Note that classification performances are directly related to how well the network gets trained, and that better training may be obtained by including a wider range of propagation models and SNR ranges. In addition, note that that the overall classification process considered in this work does not take into account any a-priori knowledge of the propagation environment. However, some type of propagation channel information, such as the general type of channel (i.e., rural or urban areas), may be available in some situations. Incorporating a-priori information will lead to a "better" training of the neural network with data selected for the specific environment of interest, resulting in improved performances.

Finally, this study did not take into account pulse shaping issues. However in practical situations, pulse shaping is commonly used prior to transmission. Adding pulse shaping and investigating the resulting effects on overall classification performances is left for further study.

APPENDIX A. AMA COST FUNCTION GRADIENT DERIVATION

Recall from Chapter IV that the output to the AMA equalizer is given by:

$$z(n)=\sum_{l=0}^{L-1} h_l(n)\cdot s(n-l), \qquad \text{(A.1)}$$

where $\underline{h}$ is the L-tap FIR equalizer weight vector at sample n, given by:

$$\underline{h}=[h_0,\ldots,h_{L-1}], \qquad \text{(A.2)}$$

and $\underline{s}$ is a portion of the input signal with length equal to the length of the filter equalizer:

$$\underline{s}=[s(n),s(n-1),\ldots s(n-L)]. \qquad \text{(A.3)}$$

Therefore, Equation (A.1) for the nth sample may be re-written in vector form as follows:

$$z=\underline{h}^T\cdot\underline{s}. \qquad \text{(A.4)}$$

Recall the AMA cost function for the nth sample is given by (Chapter IV, equation 4.7):

$$J_k = E\left\{1 - \sum_{i=1}^{M} e^{-|z-c_k(i)|^2 / 2\sigma^2}\right\}, \tag{A.5}$$

where M represents the total number of centroids for the kth constellation, $c_k(i)$ is the ith centroid of the kth constellation, and σ is a constant chosen so that:

$$e^{-|c(l)-c(i)|^2/2\sigma^2} \approx 0, \qquad \forall l \neq i. \tag{A.6}$$

The gradient of the cost function (equation A.5) is then:

$$\nabla J_k = \frac{\partial E\left\{1-\sum_{i=1}^{M} e^{-|z-c_k(i)|^2/2\sigma^2}\right\}}{\partial z}\cdot\frac{\partial z}{\partial \underline{h}} = \frac{\partial(1)}{\partial z}\cdot\frac{\partial z}{\partial \underline{h}} - \sum_{i=1}^{M}\left\{\frac{\partial\left[e^{-|z-c_k(i)|^2/2\sigma^2}\right]}{\partial z}\cdot\frac{\partial z}{\partial \underline{h}}\right\} \Rightarrow$$

$$\nabla J_k = 0 - \sum_{i=1}^{M}\left\{\left(e^{-|\underline{h}^T\underline{s}-c_k(i)|^2/2\sigma^2}\right)\frac{-2[\underline{h}^T\underline{s}-c_k(i)]^*}{2\sigma^2}\cdot\frac{\partial \underline{h}^T\underline{s}}{\partial \underline{h}}\right\} \Rightarrow \tag{A.7}$$

$$\nabla J_k = \sum_{i=1}^{M}\left\{\left(e^{-|\underline{h}^T\underline{s}-c_k(i)|^2/2\sigma^2}\right)\frac{[\underline{h}^T\underline{s}-c_k(i)]^*}{\sigma^2}\cdot\underline{s}^T\right\}.$$

APPENDIX B. DERIVATION OF MOMENT EXPRESSIONS FOR UP TO 8TH ORDER

Recall that the auto-moment for a sequence s_k was defined earlier in Chapter V as:

$$E_{s,p+q,p} = E\left[s^p (s^*)^q\right], \tag{B.1}$$

where, p and q respectively represent the number of the non conjugated and conjugated terms, respectively, and $p+q$ is the moment order.

Consider a zero-mean sequence of the form $s_k = a_k + j \cdot b_k$. For M-QAM signal types, a_k and b_k are independent, and as a result, the auto-moments are purely real [MAR98, p.169, equation B.13]. For M-FSK and M-PSK types this result does not hold, as real and imaginary sequences a_k and b_k are not independent. However, Marchand showed that for constant modulus signals such as M-FSK and M-PSK types, all moments are either zero (for odd order moments) or non-zero real quantities [MAR98, p.175, equation B.51-B.53]. Therefore, expressions for the auto-moments of modulations M-QAM, M-FSK and M-PSK can be derived easily, by applying equation (B.1) to s_k for various orders p and q and keeping the real part only. Results are shown next.

1. **Second order moments**

- $E_{S,2,2} = E[s^2(s^*)^0] = E[(a+jb)^2] \Rightarrow$
  $E_{S,2,2} = E[(a^2-b^2)]$

- $E_{S,2,1} = E[s^1(s^*)^1] = E[(a+jb)(a-jb)] \Rightarrow$
  $E_{S,2,1} = E[(a^2+b^2)]$

2. **Fourth order moments**

- $E_{S,4,4} = E[s^4(s^*)^0] = E[(a+jb)^4] = E[(a+jb)^2(a+jb)^2] \Rightarrow$
  $E_{S,4,4} = E[a^4 + 4a^3bj + 4ab^3j + 6a^2b^2j^2 + jb^4] = E[a^4 + b^4 - 6a^2b^2]$

- $E_{S,4,3} = E[s^3 s^*] = E[(a+jb)^3(a-jb)] \Rightarrow$
  $E_{S,4,3} = E[(a^3 + 3a^2bj + 3ab^2j + b^3j^3)(a-jb)] \Rightarrow$
  $E_{S,4,3} = E[a^4 + 2a^3bj - 3ab^3j - ab^3j - b^4] = E[a^4 - b^4]$

- $E_{S,4,2} = E[s^2(s^*)^2] = E[(a+jb)^2(a-jb)^2] \Rightarrow$
  $E_{S,4,2} = E[(a^2 - b^2 + 2abj)(a^2 + b^2 - 2abj)] \Rightarrow$
  $E_{S,4,2} = E[a^4 + b^4 - 2a^2b^2j^2] = E[a^4 + b^4 + 2a^2b^2]$

3. **Sixth order moments**

- $E_{S,6,6} = E[s^6(s^*)^0] = E[(a+jb)^6] = E[(a+jb)^3(a+jb)^3] \Rightarrow$
  $E_{S,6,6} = E[(a^3 + 3a^2bj + 3ab^2j^2 + b^3j^3)(a^3 + 3a^2bj + 3ab^2j^2 + b^3j^3)] \Rightarrow$
  $E_{S,6,6} = E[(a^3 + 3a^2bj - 3ab^2 - b^3j)(a^3 + 3a^2bj - 3ab^2 - b^3j)] \Rightarrow$
  $E_{S,6,6} = E[a^6 + 6a^5bj - 6a^4b^2 - 20a^3b^3j + 9a^4b^2j^2 - 6a^2b^4j^2 + 9a^2b^4 + 6ab^5j + b^6j^2] \Rightarrow$
  $E_{S,6,6} = E[a^6 - b^6 + 15a^2b^4 - 15a^4b^2]$

- $E_{S,6,5} = E[s^5 s^*] = E[(a+jb)^5(a-jb)] \Rightarrow$
  $E_{S,6,5} = E[(a^5 + 5a^4bj + 10a^3b^2j^2 + 10a^2b^3j^3 + 5ab^4j^4 + b^5j^5)(a-bj)] \Rightarrow$
  $E_{S,6,5} = E[a^6 + 4a^5bj + 5a^4b^2j^2 - 5a^2b^4j^4 - 4ab^5j^5 - b^6j^6] \Rightarrow$
  $E_{S,6,5} = E[a^6 - 5a^4b^2 - 5a^2b^4 + b^6]$

- $E_{S,6,4} = E[s^4(s^*)^2] = E[(a+jb)^4(a-jb)^2] \Rightarrow$
  $E_{S,6,4} = E[a^4 + 4a^3bj + 6a^2b^2j^2 + 4ab^3j^3 + b^4j^4)(a^2 - 2abj - b^2)] \Rightarrow$
  $E_{S,6,4} = E[a^6 + 2a^5bj - a^4b^2j^2 - 4a^3b^3j^3 - a^2b^4j^4 + 2ab^5j^5 + b^6j^6] \Rightarrow$
  $E_{S,6,4} = E[a^6 + a^4b^2 - a^2b^4 - b^6]$

- $E_{S,6,3} = E[s^3(s^*)^3] = E[(a+jb)^3(a-jb)^3] \Rightarrow$
  $E_{S,6,3} = E[a^3 + 3a^2bj + 3ab^2j^2 + b^3j^3)(a^3 - 3a^2bj + 3ab^2j^2 - b^3j^3)] \Rightarrow$
  $E_{S,6,3} = E[a^6 - 3a^4b^2j^2 + 3a^2b^4j^4 - b^6j^6] \Rightarrow$
  $E_{S,6,3} = E[a^6 + 3a^4b^2 + 3a^2b^4 + b^6]$

4. Eighth order moments

- $E_{S,8,8} = E[s^8(s^*)^0] = E[(a+jb)^8] \Rightarrow$
  $E_{S,8,8} = E[a^8 + 8a^7bj + 28a^6b^2j^2 + 56a^5b^3j^3 + 70a^4b^4j^4 + 56a^3b^5j^5 + 28a^2b^6j^6 + 8ab^7j^7 + b^8j^8] \Rightarrow$
  $E_{S,8,8} = E[a^8 - 28a^6b^2 + 70a^4b^4 - 28a^2b^6 + b^8]$

  $E_{S,8,7} = E[s^7s^*] = E[(a+jb)^7(a-jb)] \Rightarrow$
  $E_{S,8,7} = E[a^7 + 7a^6bj + 21a^5b^2j^2 + 35a^4b^3j^3 + 35a^3b^4j^4 + 21a^2b^5j^5 + 7ab^6j^6 + b^7j^7)(a-jb)] \Rightarrow$
  $E_{S,8,7} = E[a^8 + 6a^7bj + 14a^6b^2j^2 + 14a^5b^3j^3 - 14a^3b^5j^5 - 14a^2b^6j^6 - 6ab^7j^7 - b^8j^8] \Rightarrow$
  $E_{S,8,7} = E[a^8 - 14a^6b^2 + 14a^2b^6 - b^8]$

  $E_{S,8,6} = E[s^6(s^*)^2] = E[(a+jb)^6(a-jb)^2] \Rightarrow$
  $E_{S,8,6} = E[(a^6 + 6a^5bj + 15a^4b^2j^2 + 20a^3b^3j^3 + 15a^2b^4j^4 + 6ab^5j^5 + b^6j^6)(a^2 - 2abj + b^2j^2)] \Rightarrow$
  $E_{S,8,6} = E[a^8 + 4a^7bj + 4a^6b^2j^2 - 4a^5b^3j^3 - 10a^4b^4j^4 - 4a^3b^5j^5 + 4a^2b^6j^6 + 4ab^7j^7 + b^8j^8] \Rightarrow$
  $E_{S,8,6} = E[a^8 - 4a^6b^2 - 10a^4b^4 - 4a^2b^6 + b^8]$

- $$E_{S,8,5}=E[s^5(s^*)^3]=E[(a+jb)^5(a-jb)^3]\Rightarrow$$
  $$E_{S,8,5}=E[(a^5+5a^4bj+10a^3b^2j^2+10a^2b^3j^3+5ab^4j^4+b^5j^5)(a^3-3a^2bj+3ab^2j^2-b^3j^3)]\Rightarrow$$
  $$E_{S,8,5}=E[a^8+2a^7bj-2a^6b^2j^2-6a^5b^3j^3+6a^3b^5j^5+2a^2b^6j^6-2ab^7j^7-b^8j^8)]\Rightarrow$$
  $$E_{S,8,5}=E[a^8+2a^6b^2-2a^2b^6-b^8]$$

- $$E_{S,8,4}=E[s^4(s^*)^4]=E[(a+jb)^4(a-jb)^4]\Rightarrow$$
  $$E_{S,8,4}=E[(a^4+4a^3bj+6a^2b^2j^2+4ab^3j^3+b^4j^4)(a^4-4a^3bj+6a^2b^2j^2-4ab^3j^3+b^4j^4)]\Rightarrow$$
  $$E_{S,8,4}=E[a^8-4a^6b^2j^2+6a^4b^4j^4-4a^2b^6j^6+b^8j^8]\Rightarrow$$
  $$E_{S,8,4}=E[a^8+4a^6b^2+6a^4b^4+4a^2b^6+b^8]$$

APPENDIX C. PROPAGATION CHANNELS IMPULSE RESPONSES

One of the goals of the study was to simulate situations as close to reality as possible. For this reason, data taken from real world measurements were used, as opposed to artificial channel models [MPR00]. These impulse responses represent various wireless propagation channels, from mild fading to severe multipath fading situations. Figures C-1 to C-9 show the impulse responses of the channels used for the neural network training described in Chapter VII. Figures C-10 to C-15 show the impulse responses of the channels that are used during the testing phase of the overall classification scheme. All plots present the absolute value of the impulse responses in dB. One thing that worth noting is the similarity of some of these real world channels with the theoretical Rayleigh fading envelope presented in Figure III-4. However, note that there are cases where the real channels are much worse than those described by the Rayleigh fading model (Figures C-14 and C-15).

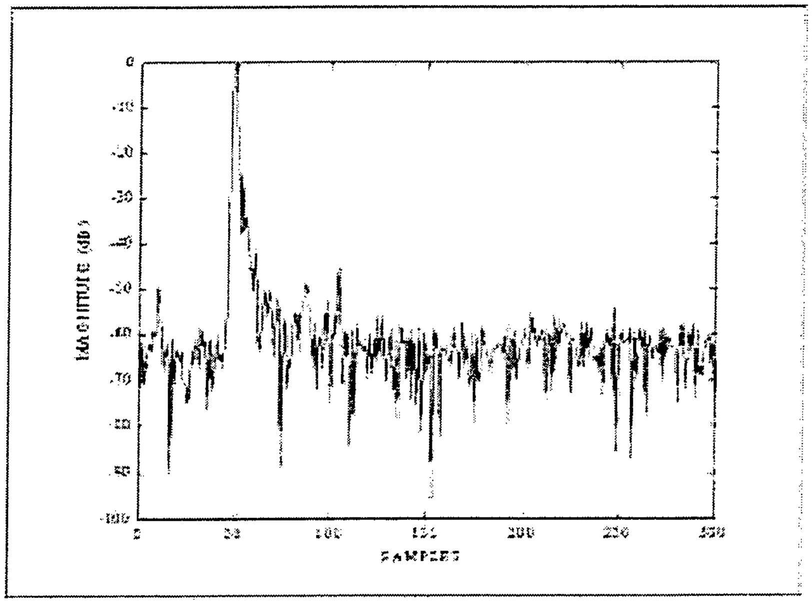


Figure C-1. Propagation channel #1.

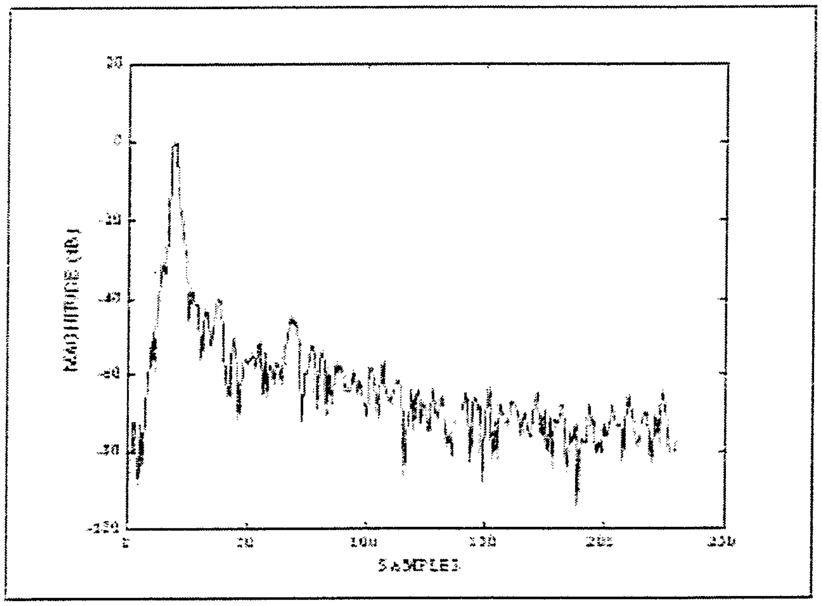


Figure C-2. Propagation channel #2.

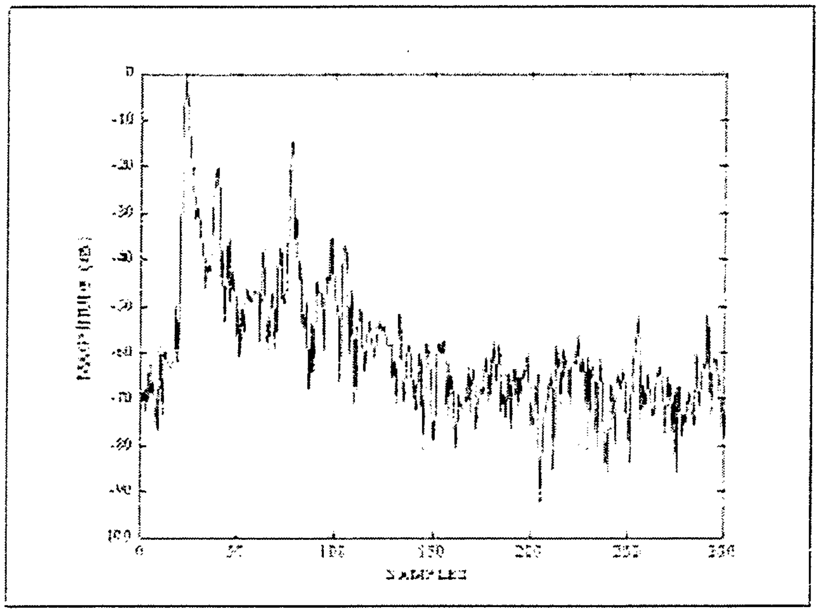


Figure C-3. Propagation channel #3.

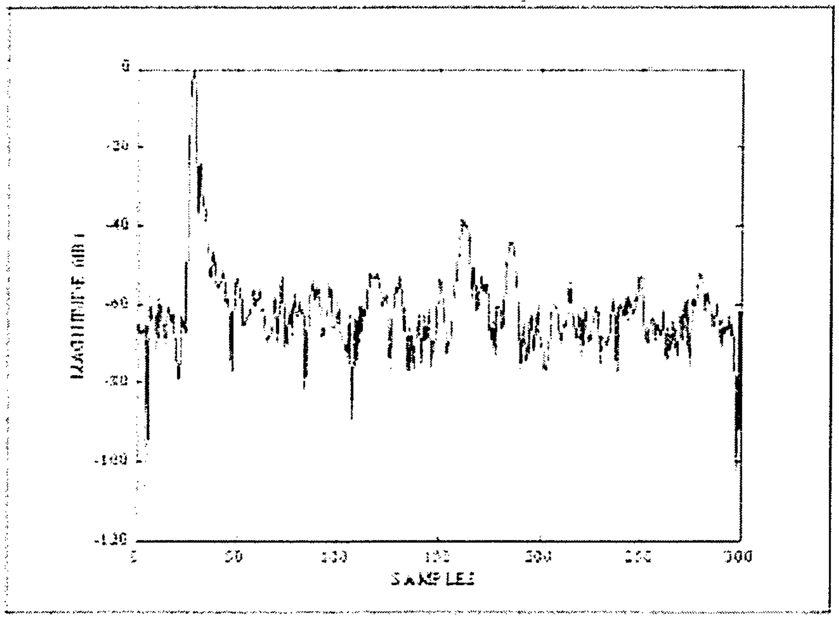


Figure C-4. Propagation channel #4.

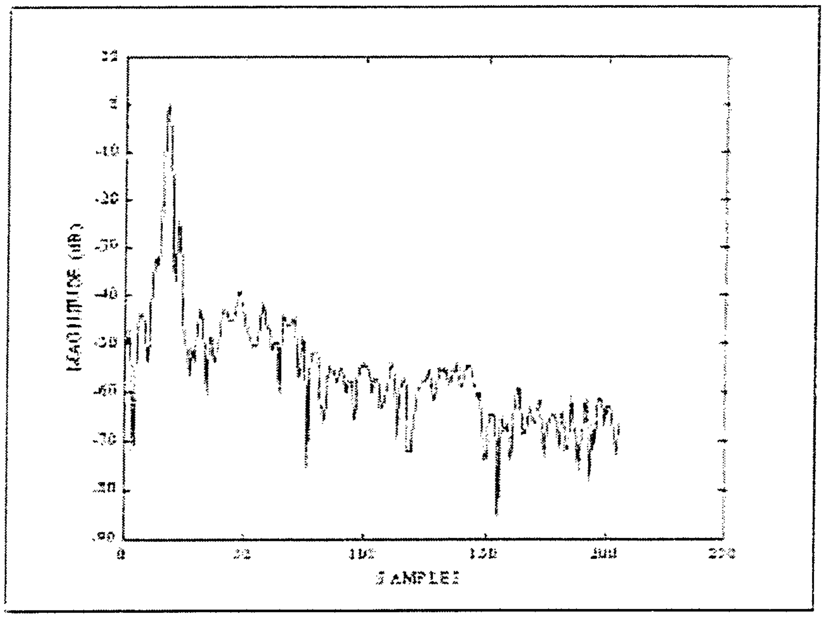


Figure C-5. Propagation channel #5.

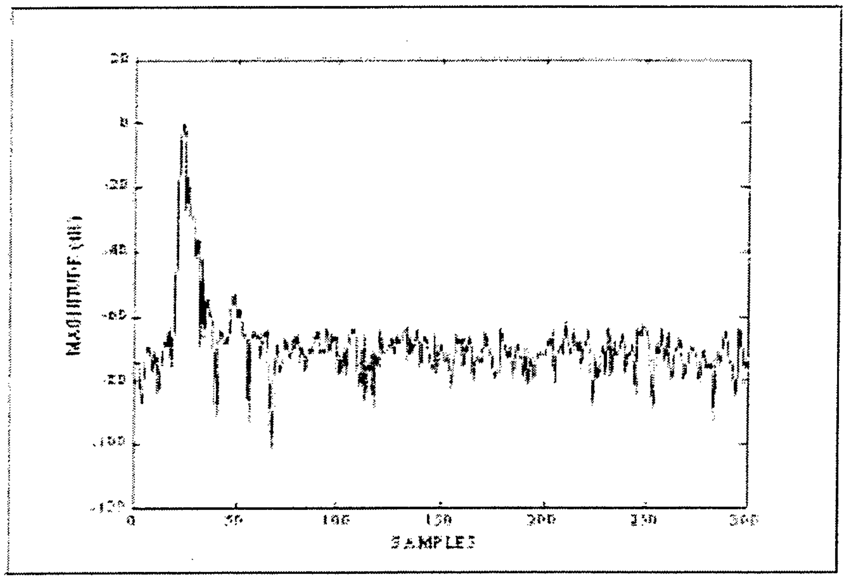


Figure C-6. Propagation channel #6.

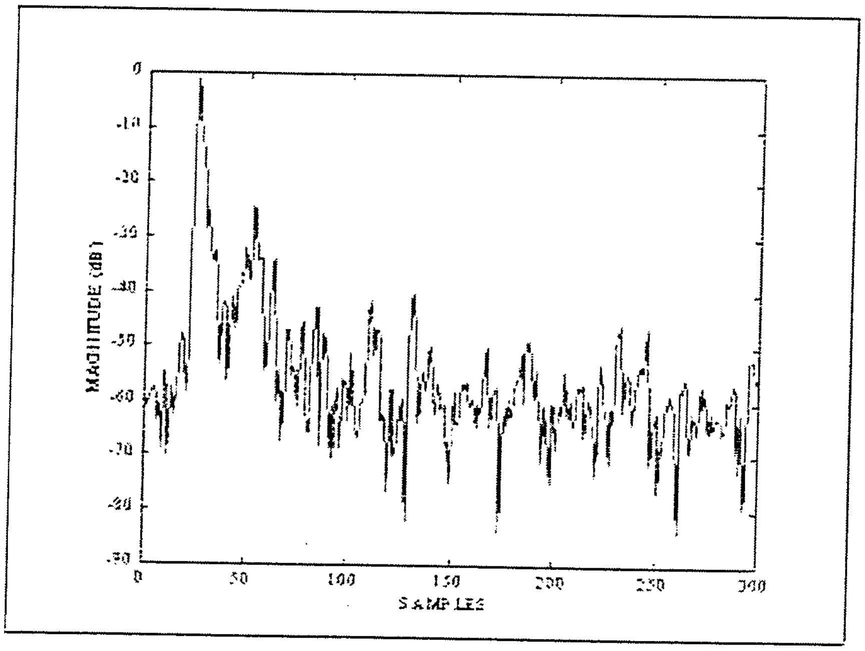


Figure C-7. Propagation channel #7.

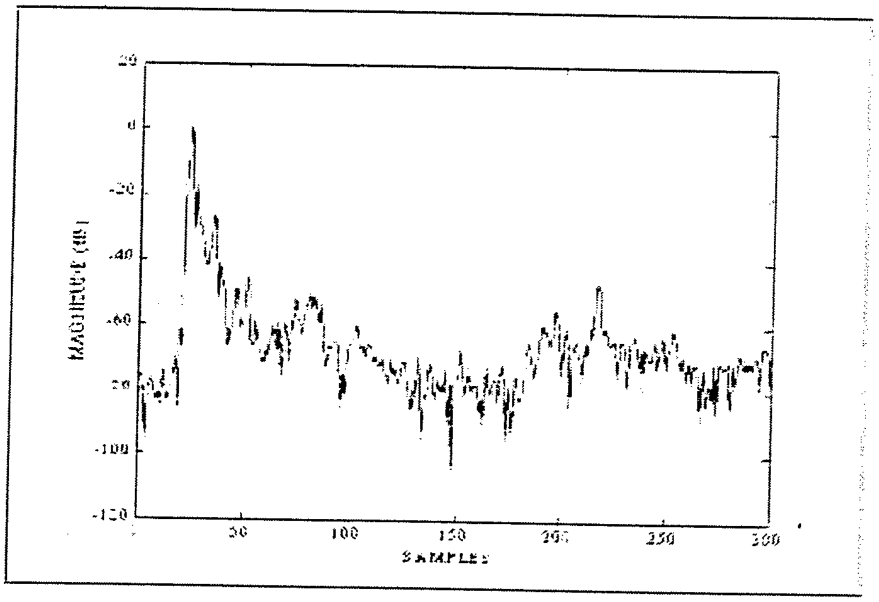


Figure C-8. Propagation channel #8.

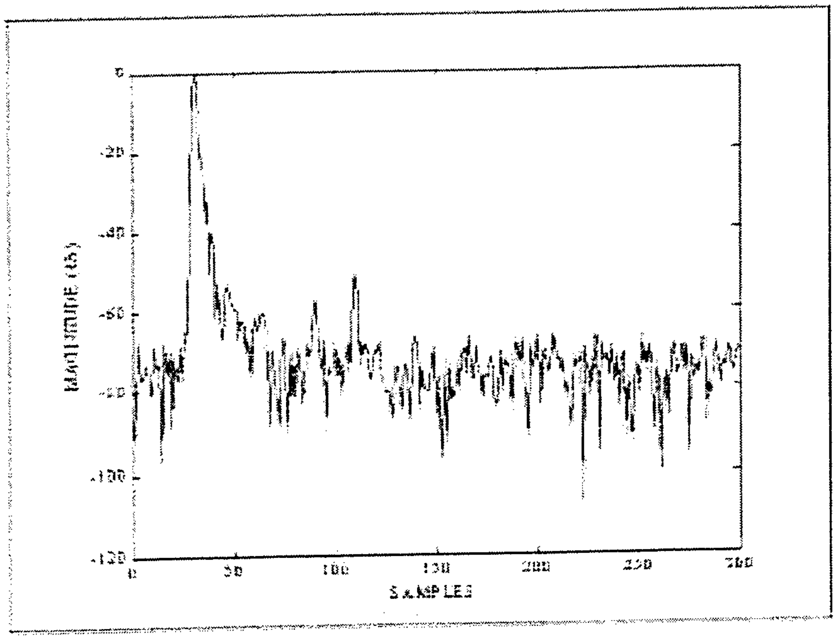


Figure C-9. Propagation channel #9.

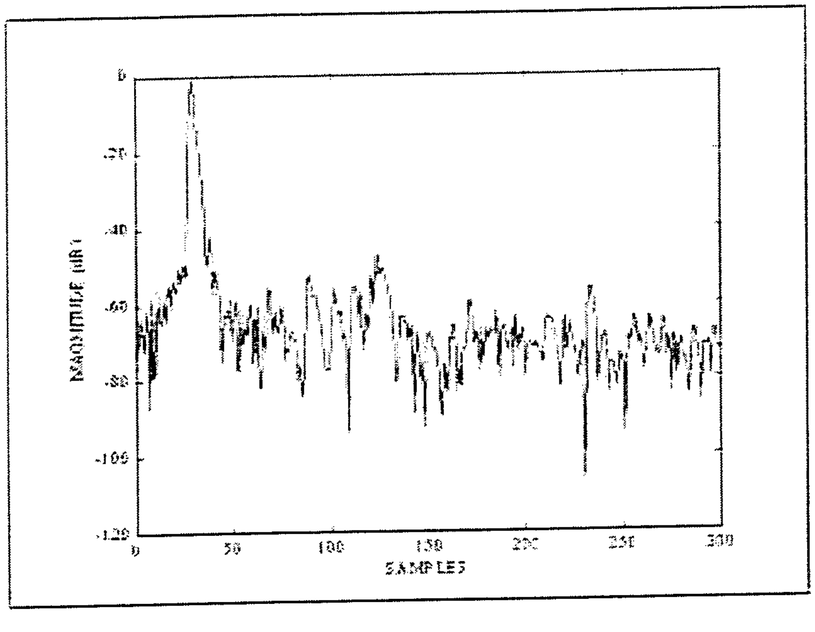


Figure C-10. Propagation channel #10.

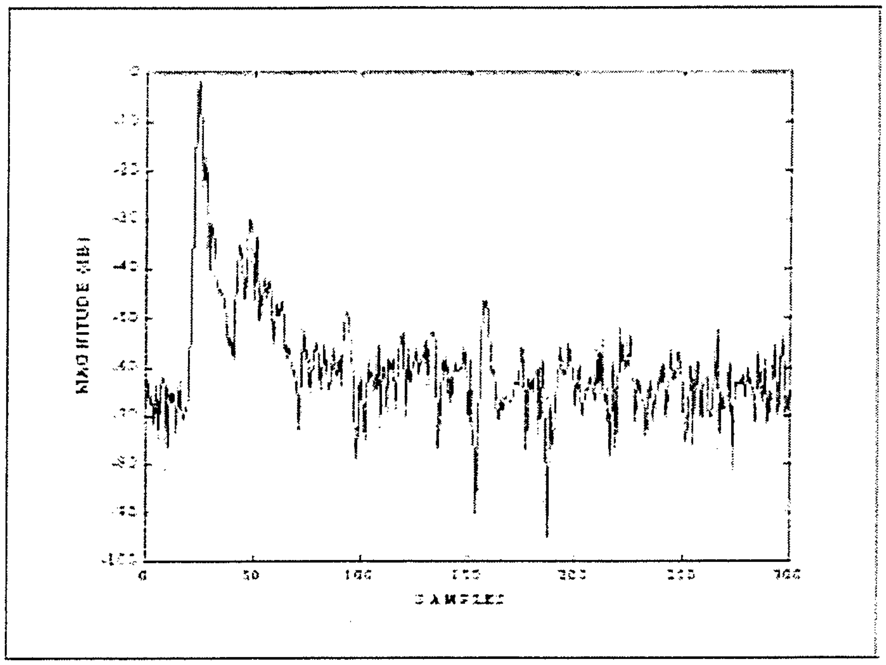


Figure C-11. Propagation channel #11.

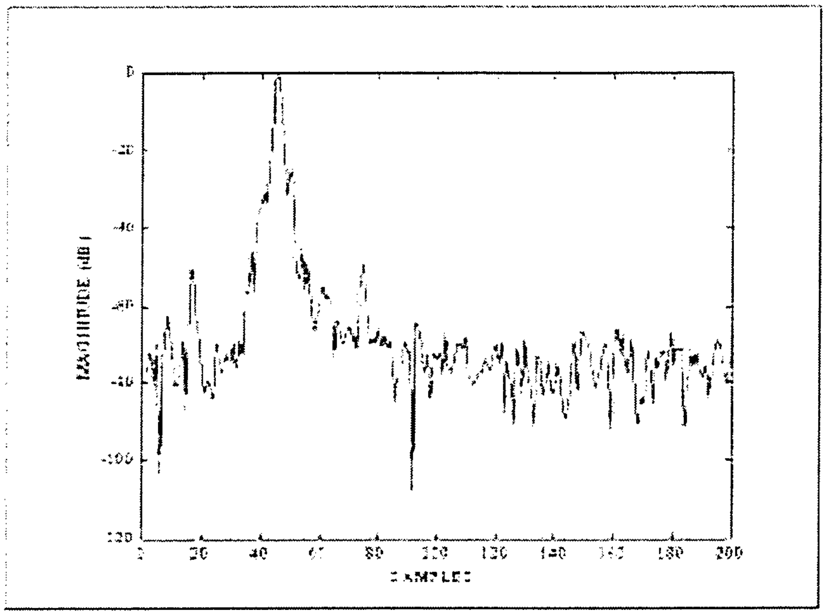


Figure C-12. Propagation channel #12.

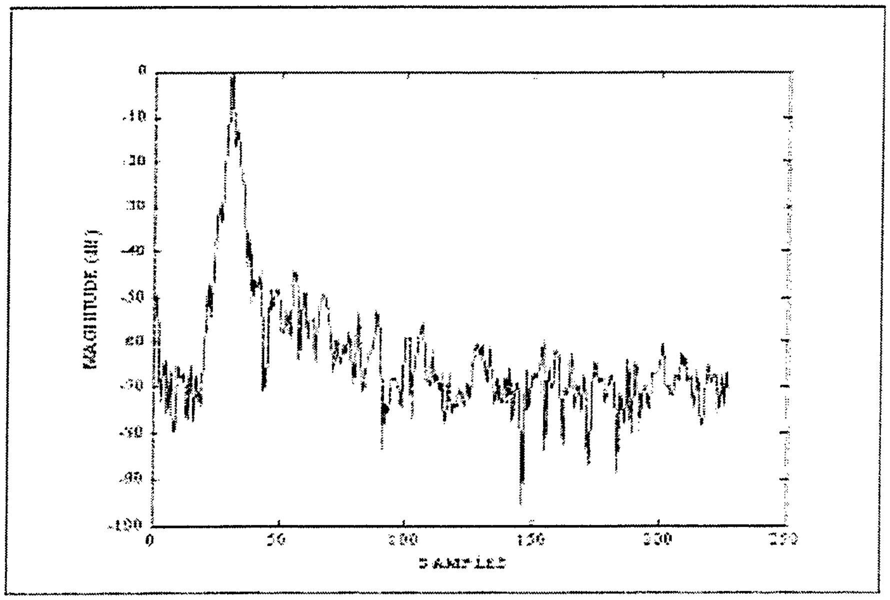


Figure C-13. Propagation channel #13.

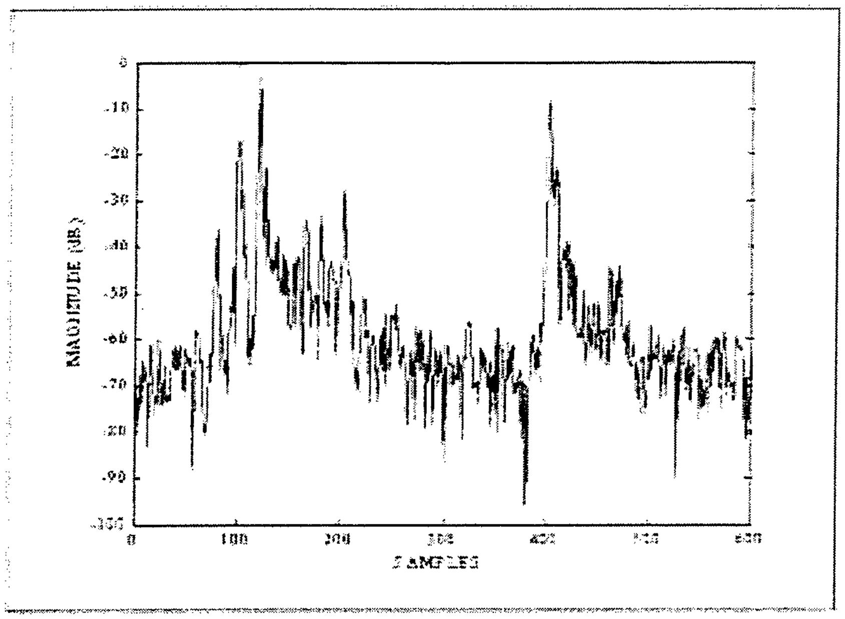


Figure C-14. Propagation channel #14.

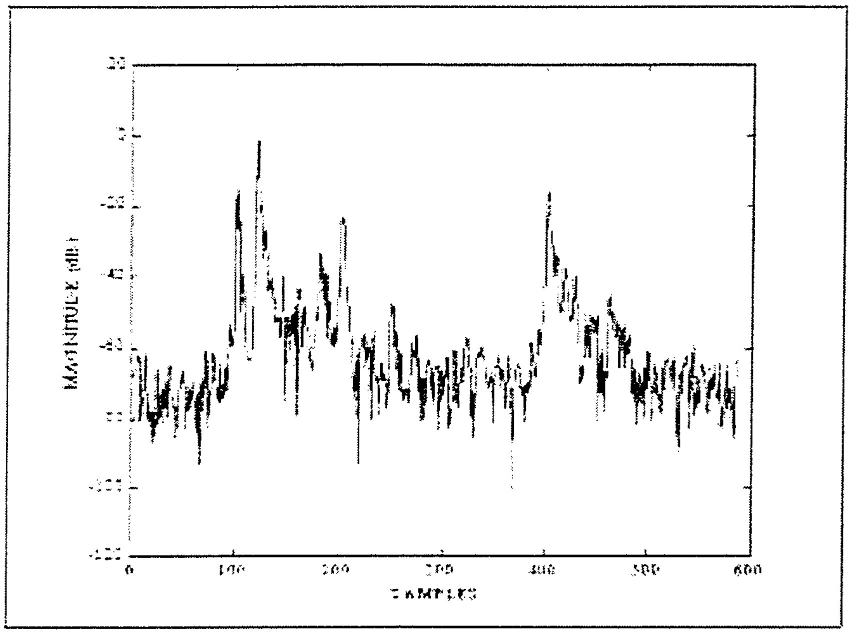


Figure C-15. Propagation channel #15.

THIS PAGE   INTENTIONALLY LEFT BLANK

APPENDIX D. MATLAB MAIN PROGRAM AND FUNCTIONS

```
%**********************************************************************
% MAIN_MENU.m
%        -Main control program
%
% Use:      This program is the user interface to the classification
scheme
%
% Input:    None
%
% Returns:  None
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%**********************************************************************
clc;clear all
disp(' ***********    M E N U   ***************')
disp('                                          ')
disp(' [1]  Create new signals for training     ')
disp(' [2]  Create training features fot the NNs ')
disp(' [3]  Train networks with excisting data ')
disp(' [4]  Start testing process  (automated) ')
disp(' [5]  Test a signal manually  (suggested....)      ')
disp('                                          ')
choice=input('Give your choice:  ')
switch choice
case 1,
   disp('   ')
   samples=input (' How many samples? ')
   CREATE_SIGNALS(samples)
case 2,
   create_moments_for_NN(dummy);
case 3,
   train_NNs(0)
case 4,
   auto_results(10)
case 5,
   disp(' WHAT SIGNAL DO YOU WANT AS THE TESTING SIGNAL ??  ')
   disp('                            ')
   disp(' [1] 2-FSK          [6] 8-PSK')
   disp(' [2] 4-FSK          [7] 16-QAM')
   disp(' [3] 8-FSK          [8] 64-QAM')
   disp(' [4] 2-PSK          [9] 256-QAM')
   disp(' [5] 4-PSK                    ')
   disp('                              ')
   choice_signal=input('Give your choice  ')
   clc
   choice_snr=input('Give the desired SNR in dB  ')
   disp('                              ')
   choice_channel=input('Which propagation model do you want [10-15]?
')
```

```
    disp('    ')
    flag=classifier(choice_signal,choice_channel,choice_snr);
    disp('The SNR was')
    choice_snr
    disp(' The channel was ')
    choice_channel
end
%                       E N D  O F  F U N C T I O N


function flag_storage=classifier(choice_signal,choice_channel,snr_db)
%*********************************************************************
% Function
%  - IMPLEMENTATION OF THE CLASSIFICATION TREE, BLOCK BY BLOCK
%
% Use: flag_storage=classifier(choice_signal,choice_channel,snr_db)
%
% Input:    choice_signal-> Allowed values 1...9 correspond to the
desired unknown signal
%                           to be classified
%           choice_channel->Allowed values 10...15 correspond to the
desired propagation channel
%                           that the unknown signal will be pased
through
%           snr_db->  The desired signal to noise ratio in db
%
% Returns:  flag_storage-> A code-value from 1...9 depending the
classifier outcome
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*********************************************************************
clc
flag_storage=0;
% Load the trained NNs for blocks 0-4 from workspace
load trained_NNs;
flag2=0;
s=0;
flag=1;
% Create fresh signals
CREATE_SIGNALS(30000);
% Load clear modulations from workspace
load testing_signals
y_2fsk=FSK_signals(:,1);
y_4fsk=FSK_signals(:,2);
y_8fsk=FSK_signals(:,3);
y_2psk=PSK_signals(:,1);
y_4psk=PSK_signals(:,2);
y_8psk=PSK_signals(:,3);
y_16qam=QAM_signals(:,1);
y_64qam=QAM_signals(:,2);
y_256qam=QAM_signals(:,3);
samples_to_keep=length(y_2fsk);
```

```
block1_input=[];
block2_input=[];
block3_input=[];
block3_input_1=[];
block3_input_2=[];
block4_input=[];
% Assign the chosen signal value to the corresponding modulation
switch choice_signal
case 1,
   x_signal=y_2fsk;
case 2,
   x_signal=y_4fsk;
case 3,
   x_signal=y_8fsk;
case 4,
   x_signal=y_2psk;
case 5,
   x_signal=y_4psk;
case 6
   x_signal=y_8psk;
case 7,
   x_signal=y_16qam;
case 8,
   x_signal=y_64qam;
case 9,
   x_signal=y_256qam;
end
% Assign the chosen channel value to the corresponding channel that is
saved in the workspace
switch choice_channel
case 10,
   load chan10;
case 11,
   load chan11;
case 12;
   load chan12;
case 13,
   load chan13;
case 14,
   load chan14;
case 15,
   load chan15
end
% Pass the unknown modulatin from the chosen propagation channel
     x_signal=filter(C,1,x_signal);
% Convert dB into a number
snr=10^(snr_db/10);
% Add white noise
 [x_signal]=addAWGN(x_signal,snr);
% Subtruct the mean and normalized the noised signal
x_signal=x_signal-mean(x_signal);
x_signal_energy=(1/samples_to_keep)*norm(x_signal,2)^2;
% * * * * * * * * * * * * * * * *  * * * * * * * * * * * * * * * * *
%                    * * *  Start the hierarchical tree   * * *
% * * * * * * * * * * * * * * * * *  * * * * * * * * * * * * * * * * * *

%                                        BLOCK #0


% Estimate the cumulant C88
mom22_x_signal=real(mom2x(x_signal,0));
mom21_x_signal=real(mom2x(x_signal,1));
mom44_x_signal=real(mom4x(x_signal,0));
mom43_x_signal=real(mom4x(x_signal,1));
mom42_x_signal=real(mom4x(x_signal,2));
mom88_x_signal=real(mom8x(x_signal,0));
mom84_x_signal=real(mom8x(x_signal,1));
criterion=cum88_module(mom88_x_signal,mom44_x_signal,mom22_x_signal,mom
84_x_signal,
mom42_x_signal,mom43_x_signal,mom21_x_signal,x_signal_energy);
% Test the network
Y = sim(net_0,criterion);
% Decide on the output
if Y<=0
   disp('we have 2-PSK')
   flag_storage=4;
   flag=10;
elseif Y>0
   disp('we have 4-PSK or 8-PSK or M-FSK or M-QAM')
   flag=1;
end
% * * * * * * * * * * * * * * * * *  * * * * * * * * * * * * * * * * * *
%                                        BLOCK #1
if flag==1
count=0;chop=3000;flag3=0;block1_input_1=[];block1_input_2=[];block1_in
put=[];
   criterion1=real(mom4x(x_signal,1));
   criterion1=criterion1/x_signal_energy^2;
   criterion2=real(mom2x(x_signal,0));
   criterion2=criterion2/x_signal_energy;
   block1_input_1=[block1_input_1;criterion1];
   block1_input_2=[block1_input_2;criterion2];
   block1_input=[block1_input_1 block1_input_2]';
   % Test the network
   Y = sim(net_1,block1_input)
   % Decide on the output
   if Y>=0.20
      flag=1;
      disp('We have M-FSK')
   elseif Y<=-0.20
      disp('We have M-FSK')
      flag=1;
   else
      disp('We have 4-PSK, or 8-PSK or M-QAM')
      flag=0;
   end
```

```
end
% * * * * * * * * * * * * * * * * *  * * * * * * * * * * * * * * * * * *
%                                           BLOCK #2
if flag==1
   count=0;
   mom65_x_signal=real(mom6x(x_signal));
   mom22_x_signal=real(mom2x(x_signal,0));
   mom43_x_signal=real(mom4x(x_signal,1));
   mom21_x_signal=real(mom2x(x_signal,1));
   mom44_x_signal=real(mom4x(x_signal,0));
criterion=cum65_module(mom65_x_signal,mom44_x_signal,mom43_x_signal,mom
22_x_signal,mom21_x_signal,x_signal_energy);
   % Test the network
   Y=sim(net_2,criterion)
   % Decide on the output
if Y<=0.3
   disp('we have 2-FSK')
   flag_storage=1;
   flag=10;
elseif Y>0.3;
   disp('we have 4-FSK or 8-FSK')
   flag=2;
end
% * * * * * * * * * * * * * * * * *  * * * * * * * * * * * * * * * * * *
%                                           BLOCK #3
if flag==2
   mom65_x_signal=real(mom6x(x_signal));
   criterion=mom65_x_signal/x_signal_energy^3;
   % Test the network
   Y=sim(net_3,criterion)
   % Decide on the output
   if Y>0
      disp('we have 4-FSK')
      flag_storage=2;
      flag=10;
   else
      disp('we have 8-FSK')
      flag_storage=3;
      flag=10;
   end
end
end
% * * * * * * * * * * * * * * * * *  * * * * * * * * * * * * * * * * * *
%                                           BLOCK #4
if flag==0
   mom65_x_signal=real(mom6x(x_signal));
   mom22_x_signal=real(mom2x(x_signal,0));
   mom43_x_signal=real(mom4x(x_signal,1));
   mom21_x_signal=real(mom2x(x_signal,1));
   mom44_x_signal=real(mom4x(x_signal,0));
criterion=cum65_module(mom65_x_signal,mom44_x_signal,mom43_x_signal,mom
22_x_signal,mom21_x_signal,x_signal_energy);
   % Test the network
```

```
    Y=sim(net_4,criterion)
    % Decide on the output
    if Y>=0.3
       flag_storage=6;
       flag=10;
       disp('We have 8-PSK')
    elseif Y<=-0.3
       flag=0;
       disp('We have M-QAM')
    else
       disp('We have  4-PSK')
       flag_storage=5;
       flag=10;
    end
end
% * * * * * * * * * * * * * * * *  * * * * * * * * * * * * * * * * * *
%                                        BLOCK #5
if flag==0
   % Call the AMA function to separate the M-QAMs
[flag_storage,CF1,CF2,CF3,final_1,final_2,final_3]=ama_function(x_signa
l);
   if flag_storage==100
      % If AMA cannot make a decision, call it once more
[flag_storage,CF1,CF2,CF3,final_1,final_2,final_3]=ama_function(x_signa
l);
   end
end
return
%                      E N D  O F  F U N C T I O N


function auto_results(choice_channel)
%***********************************************************************
% Function
%  - RUNS AUTOMATICALLY THE CLASSIFICATION TREE FOR DIFFERENT
SNRS,TRIALS AND
%    PROPAGATION CHANNELS
%
% Use: auto_results(dummy)
%
% Input:    dummy-> A dummy variable
%
% Returns:  The confusion matrixes of all simulations.
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%***********************************************************************
clear all;
% Declaration of confusion matrixes
confusion=[];
confusion_cascade=[];
% CHOICE OF THE PROPAGATION SIGNAL
% *****************
```

```
 choice_channel=10;
% *****************
% Declaration of the simulation SNRs and trials
snrdb=[20 17 14 11 8 5 2];
trials=50;
% First loop is for the different SNRs
for count=1:7
   snr_db=snrdb(count);
   % Second loop is for the different types of modulations
   for choice_signal=1:9
      vector=zeros(trials,9);
      % Third loop is for the different trials
      for trial_count=1:trials
         flag_storage=classifier(choice_signal,choice_channel,snr_db)
         if flag_storage==100
            flag_storage=8;
         elseif flag_storage==101
            flag_storage=7;
         elseif flag_storage==102
            flag_storage=7;
         end
         vector(trial_count,flag_storage)=1;
      end
      row_vector=sum(vector);
      confusion=[confusion;row_vector];
   end
   confusion_cascade(:,:,count)=confusion;
   confusion=[];
save temporary_results_channel_10 confusion_cascade count choice_signal
trial_count;
end
% Save the confusion matrixes into the workspace
save results_channel_10 confusion_cascade
return
%                       E N D   O F   F U N C T I O N


function create_moments_for_NN(dummy);

%************************************************************************
% Function
%  - Creates the training data-set for the NNs of blocks 0 to 4
%
% Use: create_moments_for_NN(dummy)
%
% Input:    None
%
% Returns:  The training dataset is saved to workspace
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%************************************************************************
```

```
% Load the clean signals
load testing_signals
samples_to_keep=length(FSK_signals);
% Initialization of function's internal variables
SUM_mom43=[];SUM_mom65=[];SUM_cum65=[];SUM_cur=[];SUM_mom44=[];SUM_mom8
6=[];SUM_mom42=[];
SUM_cum44=[];SUM_cum88=[];SUM_mom22=[];SUM_snr=[];SUM_cum84=[];SUM_mom8
4=[];
FSK_signals=FSK_signals(1:samples_to_keep,:);
PSK_signals=PSK_signals(1:samples_to_keep,:);
QAM_signals=QAM_signals(1:samples_to_keep,:);
% The first loop is for the different channels
for loop3=2:10
   snr_db=20;
% The second loop is for the different SNRs
   for loop1=1:20
      % Define the desired SNR
      snr_db=snr_db-1;
      snr=10^(snr_db/10);
% The third loop is for the number of samples per snr
for loop2=1:10
   y_2fsk=FSK_signals(:,1);
   y_4fsk=FSK_signals(:,2);
   y_8fsk=FSK_signals(:,3);
   y_2psk=PSK_signals(:,1);
   y_4psk=PSK_signals(:,2);
   y_8psk=PSK_signals(:,3);
   y_16qam=QAM_signals(:,1);
   y_64qam=QAM_signals(:,2);
   y_256qam=QAM_signals(:,3);
   if  loop3==2
      load chan1
      y_2fsk=filter(C,1,y_2fsk);
      y_4fsk=filter(C,1,y_4fsk);
      y_8fsk=filter(C,1,y_8fsk);
      y_2psk=filter(C,1,y_2psk);
      y_4psk=filter(C,1,y_4psk);
      y_8psk=filter(C,1,y_8psk);
      y_16qam=filter(C,1,y_16qam);
      y_64qam=filter(C,1,y_64qam);
      y_256qam=filter(C,1,y_256qam);
   elseif loop3==3
      load chan2
      y_2fsk=filter(C,1,y_2fsk);
      y_4fsk=filter(C,1,y_4fsk);
      y_8fsk=filter(C,1,y_8fsk);
      y_2psk=filter(C,1,y_2psk);
      y_4psk=filter(C,1,y_4psk);
      y_8psk=filter(C,1,y_8psk);
      y_16qam=filter(C,1,y_16qam);
      y_64qam=filter(C,1,y_64qam);
      y_256qam=filter(C,1,y_256qam);
   elseif loop3==4
      load chan3
   y_2fsk=filter(C,1,y_2fsk);
   y_4fsk=filter(C,1,y_4fsk);
   y_8fsk=filter(C,1,y_8fsk);
   y_2psk=filter(C,1,y_2psk);
   y_4psk=filter(C,1,y_4psk);
   y_8psk=filter(C,1,y_8psk);
   y_16qam=filter(C,1,y_16qam);
   y_64qam=filter(C,1,y_64qam);
   y_256qam=filter(C,1,y_256qam);
elseif loop3==5
   load chan4
   y_2fsk=filter(C,1,y_2fsk);
   y_4fsk=filter(C,1,y_4fsk);
   y_8fsk=filter(C,1,y_8fsk);
   y_2psk=filter(C,1,y_2psk);
   y_4psk=filter(C,1,y_4psk);
   y_8psk=filter(C,1,y_8psk);
   y_16qam=filter(C,1,y_16qam);
   y_64qam=filter(C,1,y_64qam);
   y_256qam=filter(C,1,y_256qam);
elseif loop3==6
   load chan5
   y_2fsk=filter(C,1,y_2fsk);
   y_4fsk=filter(C,1,y_4fsk);
   y_8fsk=filter(C,1,y_8fsk);
   y_2psk=filter(C,1,y_2psk);
   y_4psk=filter(C,1,y_4psk);
   y_8psk=filter(C,1,y_8psk);
   y_16qam=filter(C,1,y_16qam);
   y_64qam=filter(C,1,y_64qam);
   y_256qam=filter(C,1,y_256qam);
elseif loop3==7
   load chan6
   y_2fsk=filter(C,1,y_2fsk);
   y_4fsk=filter(C,1,y_4fsk);
   y_8fsk=filter(C,1,y_8fsk);
   y_2psk=filter(C,1,y_2psk);
   y_4psk=filter(C,1,y_4psk);
   y_8psk=filter(C,1,y_8psk);
   y_16qam=filter(C,1,y_16qam);
   y_64qam=filter(C,1,y_64qam);
   y_256qam=filter(C,1,y_256qam);
elseif loop3==8
   load chan7
   y_2fsk=filter(C,1,y_2fsk);
   y_4fsk=filter(C,1,y_4fsk);
   y_8fsk=filter(C,1,y_8fsk);
   y_2psk=filter(C,1,y_2psk);
   y_4psk=filter(C,1,y_4psk);
   y_8psk=filter(C,1,y_8psk);
   y_16qam=filter(C,1,y_16qam);
   y_64qam=filter(C,1,y_64qam);
   y_256qam=filter(C,1,y_256qam);
elseif loop3==9
```

```
 load chan8
 y_2fsk=filter(C,1,y_2fsk);
 y_4fsk=filter(C,1,y_4fsk);
 y_8fsk=filter(C,1,y_8fsk);
 y_2psk=filter(C,1,y_2psk);
 y_4psk=filter(C,1,y_4psk);
 y_8psk=filter(C,1,y_8psk);
 y_16qam=filter(C,1,y_16qam);
 y_64qam=filter(C,1,y_64qam);
 y_256qam=filter(C,1,y_256qam);
 elseif loop3==10
 load chan9
 y_2fsk=filter(C,1,y_2fsk);
 y_4fsk=filter(C,1,y_4fsk);
 y_8fsk=filter(C,1,y_8fsk);
 y_2psk=filter(C,1,y_2psk);
 y_4psk=filter(C,1,y_4psk);
 y_8psk=filter(C,1,y_8psk);
 y_16qam=filter(C,1,y_16qam);
 y_64qam=filter(C,1,y_64qam);
 y_256qam=filter(C,1,y_256qam);
 end
 % Add white noise to form various SNRs
  [y_2fsk]=addAWGN(y_2fsk,snr);
  [y_4fsk]=addAWGN(y_4fsk,snr);
  [y_8fsk]=addAWGN(y_8fsk,snr);
  [y_2psk]=addAWGN(y_2psk,snr);
  [y_4psk]=addAWGN(y_4psk,snr);
  [y_8psk]=addAWGN(y_8psk,snr);
  [y_16qam]=addAWGN(y_16qam,snr);
  [y_64qam]=addAWGN(y_64qam,snr);
  [y_256qam]=addAWGN(y_256qam,snr);
 % Find the kurtosis of all signals
 cur_2fsk=kurtosis(y_2fsk);
 cur_4fsk=kurtosis(y_4fsk);
 cur_8fsk=kurtosis(y_8fsk);
 cur_2psk=kurtosis(y_2psk);
 cur_4psk=kurtosis(y_4psk);
 cur_8psk=kurtosis(y_8psk);
 cur_16qam=kurtosis(y_16qam);
 cur_64qam=kurtosis(y_64qam);
 cur_256qam=kurtosis(y_256qam);
 acur=[cur_2fsk cur_4fsk cur_8fsk cur_2psk cur_4psk cur_8psk cur_16qam
 cur_64qam cur_256qam];
 SUM_cur=[SUM_cur;acur];
 acur=[];
 SUM_snr=[SUM_snr;snr_db];
 % Subtract the mean of all signals
 y_2psk=y_2psk-mean(y_2psk);
 y_4psk=y_4psk-mean(y_4psk);
 y_8psk=y_8psk-mean(y_8psk);
 y_2fsk=y_2fsk-mean(y_2fsk);
 y_4fsk=y_4fsk-mean(y_4fsk);
 y_8fsk=y_8fsk-mean(y_8fsk);
```

```
y_16qam=y_16qam-mean(y_16qam);
y_64qam=y_64qam-mean(y_64qam);
y_256qam=y_256qam-mean(y_256qam);
% Find the energy of all signals
y_2psk_energy=(1/samples_to_keep)*norm(y_2psk,2)^2;
y_4psk_energy=(1/samples_to_keep)*norm(y_4psk,2)^2;
y_8psk_energy=(1/samples_to_keep)*norm(y_8psk,2)^2;
y_2fsk_energy=(1/samples_to_keep)*norm(y_2fsk,2)^2;
y_4fsk_energy=(1/samples_to_keep)*norm(y_4fsk,2)^2;
y_8fsk_energy=(1/samples_to_keep)*norm(y_8fsk,2)^2;
y_16qam_energy=(1/samples_to_keep)*norm(y_16qam,2)^2;
y_64qam_energy=(1/samples_to_keep)*norm(y_64qam,2)^2;
y_256qam_energy=(1/samples_to_keep)*norm(y_256qam,2)^2;
% Evaluate all the higher order statistics
mom43_y_2fsk =real(mom4x( y_2fsk,1));
mom43_y_2fsk_n=mom43_y_2fsk/ y_2fsk_energy^2;
mom43_y_4fsk =real(mom4x( y_4fsk,1));
mom43_y_4fsk_n=mom43_y_4fsk/ y_4fsk_energy^2;
mom43_y_8fsk =real(mom4x( y_8fsk,1));
mom43_y_8fsk_n=mom43_y_8fsk/ y_8fsk_energy^2;
mom43_y_2psk =real(mom4x( y_2psk,1));
mom43_y_2psk_n=mom43_y_2psk/ y_2psk_energy^2;
mom43_y_4psk =real(mom4x( y_4psk,1));
mom43_y_4psk_n=mom43_y_4psk/ y_4psk_energy^2;
mom43_y_8psk =real(mom4x( y_8psk,1));
mom43_y_8psk_n=mom43_y_8psk/ y_8psk_energy^2;
mom43_y_16qam =real(mom4x( y_16qam,1));
mom43_y_16qam_n=mom43_y_16qam/ y_16qam_energy^2;
mom43_y_64qam =real(mom4x( y_64qam,1));
mom43_y_64qam_n=mom43_y_64qam/ y_64qam_energy^2;
mom43_y_256qam =real(mom4x( y_256qam,1));
mom43_y_256qam_n=mom43_y_256qam/ y_256qam_energy^2;
a43=[mom43_y_2fsk_n mom43_y_4fsk_n mom43_y_8fsk_n mom43_y_2psk_n
mom43_y_4psk_n mom43_y_8psk_n mom43_y_16qam_n mom43_y_64qam_n
mom43_y_256qam_n];
SUM_mom43=[SUM_mom43;a43];
a43=[];
% MOM42
mom42_y_2fsk =real(mom4x( y_2fsk,2));
mom42_y_2fsk_n=mom42_y_2fsk/ y_2fsk_energy^2;
mom42_y_4fsk =real(mom4x( y_4fsk,2));
mom42_y_4fsk_n=mom42_y_4fsk/ y_4fsk_energy^2;
mom42_y_8fsk =real(mom4x( y_8fsk,2));
mom42_y_8fsk_n=mom42_y_8fsk/ y_8fsk_energy^2;
mom42_y_2psk =real(mom4x( y_2psk,2));
mom42_y_2psk_n=mom42_y_2psk/ y_2psk_energy^2;
mom42_y_4psk =real(mom4x( y_4psk,2));
mom42_y_4psk_n=mom42_y_4psk/ y_4psk_energy^2;
mom42_y_8psk =real(mom4x( y_8psk,2));
mom42_y_8psk_n=mom42_y_8psk/ y_8psk_energy^2;
mom42_y_16qam =real(mom4x( y_16qam,2));
mom42_y_16qam_n=mom42_y_16qam/ y_16qam_energy^2;
mom42_y_64qam =real(mom4x( y_64qam,2));
mom42_y_64qam_n=mom42_y_64qam/ y_64qam_energy^2;
```

```
mom42_y_256qam =real(mom4x( y_256qam,2));
mom42_y_256qam_n=mom42_y_256qam/ y_256qam_energy^2;
a42=[mom42_y_2fsk_n mom42_y_4fsk_n mom42_y_8fsk_n mom42_y_2psk_n
mom42_y_4psk_n mom42_y_8psk_n mom42_y_16qam_n mom42_y_64qam_n
mom42_y_256qam_n];
SUM_mom42=[SUM_mom42;a42];
a42=[];
%MOM65
mom65_y_2fsk =real(mom6x( y_2fsk));
mom65_y_2fsk_n=mom65_y_2fsk/y_2fsk_energy^3;
mom65_y_4fsk =real(mom6x( y_4fsk));
mom65_y_4fsk_n=mom65_y_4fsk/y_4fsk_energy^3;
mom65_y_8fsk =real(mom6x( y_8fsk));
mom65_y_8fsk_n=mom65_y_8fsk/y_8fsk_energy^3;
mom65_y_2psk =real(mom6x( y_2psk));
mom65_y_2psk_n=mom65_y_2psk/y_2psk_energy^3;
mom65_y_4psk =real(mom6x( y_4psk));
mom65_y_4psk_n=mom65_y_4psk/y_4psk_energy^3;
mom65_y_8psk =real(mom6x( y_8psk));
mom65_y_8psk_n=mom65_y_8psk/y_8psk_energy^3;
mom65_y_16qam =real(mom6x( y_16qam));
mom65_y_16qam_n=mom65_y_16qam/y_16qam_energy^3;
mom65_y_64qam =real(mom6x( y_64qam ));
mom65_y_64qam_n=mom65_y_64qam/y_64qam_energy^3;
mom65_y_256qam =real(mom6x( y_256qam));
mom65_y_256qam_n=mom65_y_256qam/y_256qam_energy^3;
a65=[mom65_y_2fsk_n mom65_y_4fsk_n mom65_y_8fsk_n mom65_y_2psk_n
mom65_y_4psk_n mom65_y_8psk_n mom65_y_16qam_n mom65_y_64qam_n
mom65_y_256qam_n];
SUM_mom65=[SUM_mom65;a65];
a65=[];
% mom 84
mom84_y_2fsk =real(mom8x(y_2fsk,1));
mom84_y_4fsk =real(mom8x(y_4fsk,1));
mom84_y_8fsk =real(mom8x(y_8fsk,1));
mom84_y_2psk =real(mom8x(y_2psk,1));
mom84_y_4psk =real(mom8x(y_4psk,1));
mom84_y_8psk =real(mom8x(y_8psk,1));
mom84_y_16qam =real(mom8x(y_16qam,1));
mom84_y_64qam =real(mom8x(y_64qam,1));
mom84_y_256qam=real(mom8x(y_256qam,1));
mom84_y_2fsk_n=mom84_y_2fsk/y_2fsk_energy^4;
mom84_y_4fsk_n=mom84_y_4fsk/y_4fsk_energy^4;
mom84_y_8fsk_n=mom84_y_8fsk/y_8fsk_energy^4;
mom84_y_2psk_n=mom84_y_2psk/y_2psk_energy^4;
mom84_y_4psk_n=mom84_y_4psk/y_4psk_energy^4;
mom84_y_8psk_n=mom84_y_8psk/y_8psk_energy^4;
mom84_y_16qam_n=mom84_y_16qam/y_16qam_energy^4;
mom84_y_64qam_n=mom84_y_64qam/y_64qam_energy^4;
mom84_y_256qam_n=mom84_y_256qam/y_256qam_energy^4;
m84=[mom84_y_2fsk_n mom84_y_4fsk_n mom84_y_8fsk_n mom84_y_2psk_n
mom84_y_4psk_n mom84_y_8psk_n mom84_y_16qam_n mom84_y_64qam_n
mom84_y_256qam_n];
```

```
SUM_mom84=[SUM_mom84;m84];
m84=[];
%mom86
mom86_y_2fsk =real(mom8x(y_2fsk,2));
mom86_y_4fsk =real(mom8x(y_4fsk,2));
mom86_y_8fsk =real(mom8x(y_8fsk,2));
mom86_y_2psk =real(mom8x(y_2psk,2));
mom86_y_4psk =real(mom8x(y_4psk,2));
mom86_y_8psk =real(mom8x(y_8psk,2));
mom86_y_16qam =real(mom8x(y_16qam,2));
mom86_y_64qam =real(mom8x(y_64qam,2));
mom86_y_256qam=real(mom8x(y_256qam,2));
mom86_y_2fsk_n=mom86_y_2fsk/y_2fsk_energy^4;
mom86_y_4fsk_n=mom86_y_4fsk/y_4fsk_energy^4;
mom86_y_8fsk_n=mom86_y_8fsk/y_8fsk_energy^4;
mom86_y_2psk_n=mom86_y_2psk/y_2psk_energy^4;
mom86_y_4psk_n=mom86_y_4psk/y_4psk_energy^4;
mom86_y_8psk_n=mom86_y_8psk/y_8psk_energy^4;
mom86_y_16qam_n=mom86_y_16qam/y_16qam_energy^4;
mom86_y_64qam_n=mom86_y_64qam/y_64qam_energy^4;
mom86_y_256qam_n=mom86_y_256qam/y_256qam_energy^4;
m86=[mom86_y_2fsk_n mom86_y_4fsk_n mom86_y_8fsk_n mom86_y_2psk_n
mom86_y_4psk_n mom86_y_8psk_n mom86_y_16qam_n mom86_y_64qam_n
mom86_y_256qam_n];
SUM_mom86=[SUM_mom86;m86];
m86=[];
% C44
cum44_y_2fsk
=real(cum4x((y_2fsk),conj(y_2fsk),y_2fsk,conj(y_2fsk),0,samples_to_keep
,0,'biased'));
cum44_y_2fsk_n=cum44_y_2fsk/ y_2fsk_energy^2;
cum44_y_4fsk
=real(cum4x((y_4fsk),conj(y_4fsk),y_4fsk,conj(y_4fsk),0,samples_to_keep
,0,'biased'));
cum44_y_4fsk_n=cum44_y_4fsk/ y_4fsk_energy^2;
cum44_y_8fsk
=real(cum4x((y_8fsk),conj(y_8fsk),y_8fsk,conj(y_8fsk),0,samples_to_keep
,0,'biased'));
cum44_y_8fsk_n=cum44_y_8fsk/ y_8fsk_energy^2;
cum44_y_2psk
=real(cum4x((y_2psk),conj(y_2psk),y_2psk,conj(y_2psk),0,samples_to_keep
,0,'biased'));
cum44_y_2psk_n=cum44_y_2psk/ y_2psk_energy^2;
cum44_y_4psk
=real(cum4x((y_4psk),conj(y_4psk),y_4psk,conj(y_4psk),0,samples_to_keep
,0,'biased'));
cum44_y_4psk_n=cum44_y_4psk/ y_4psk_energy^2;
cum44_y_8psk
=real(cum4x((y_8psk),conj(y_8psk),y_8psk,conj(y_8psk),0,samples_to_keep
,0,'biased'));
cum44_y_8psk_n=cum44_y_8psk/ y_8psk_energy^2;
cum44_y_16qam
=real(cum4x((y_16qam),conj(y_16qam),y_16qam,conj(y_16qam),0,samples_to_
keep,0,'biased'));
```

```
cum44_y_16qam_n=cum44_y_16qam/y_16qam_energy^2;
cum44_y_64qam
=real(cum4x((y_64qam),conj(y_64qam),y_64qam,conj(y_64qam),0,samples_to_
keep,0,'biased'));
cum44_y_64qam_n=cum44_y_64qam/y_64qam_energy^2;
cum44_y_256qam
=real(cum4x((y_256qam),conj(y_256qam),y_256qam,conj(y_256qam),0,samples
_to_keep,0,'biased'));
cum44_y_256qam_n=cum44_y_256qam/y_256qam_energy^2;
a44=[cum44_y_2fsk_n cum44_y_4fsk_n cum44_y_8fsk_n cum44_y_2psk_n
cum44_y_4psk_n cum44_y_8psk_n cum44_y_16qam_n cum44_y_64qam_n
cum44_y_256qam_n];
SUM_cum44=[SUM_cum44;a44];
a44=[];
% C88
mom88_y_2fsk =real(mom8x(y_2fsk,0));
mom88_y_4fsk =real(mom8x(y_4fsk,0));
mom88_y_8fsk =real(mom8x(y_8fsk,0));
mom88_y_2psk =real(mom8x(y_2psk,0));
mom88_y_4psk =real(mom8x(y_4psk,0));
mom88_y_8psk =real(mom8x(y_8psk,0));
mom88_y_16qam =real(mom8x(y_16qam,0));
mom88_y_64qam =real(mom8x(y_64qam,0));
mom88_y_256qam=real(mom8x(y_256qam,0));
mom44_y_2fsk =real(mom4x(y_2fsk,0));
mom44_y_4fsk =real(mom4x(y_4fsk,0));
mom44_y_8fsk =real(mom4x(y_8fsk,0));
mom44_y_2psk =real(mom4x(y_2psk,0));
mom44_y_4psk =real(mom4x(y_4psk,0));
mom44_y_8psk =real(mom4x(y_8psk,0));
mom44_y_16qam =real(mom4x(y_16qam,0));
mom44_y_64qam =real(mom4x(y_64qam,0));
mom44_y_256qam =real(mom4x(y_256qam,0));
mom22_y_2fsk =real(mom2x(y_2fsk,0));
mom22_y_4fsk =real(mom2x(y_4fsk,0));
mom22_y_8fsk =real(mom2x(y_8fsk,0));
mom22_y_2psk =real(mom2x(y_2psk,0));
mom22_y_4psk =real(mom2x(y_4psk,0));
mom22_y_8psk =real(mom2x(y_8psk,0));
mom22_y_16qam =real(mom2x(y_16qam,0));
mom22_y_64qam =real(mom2x(y_64qam,0));
mom22_y_256qam =real(mom2x(y_256qam,0));
cum88_y_2fsk
=real(cum8x(mom88_y_2fsk,mom44_y_2fsk,mom22_y_2fsk,0,0,0,0,0));
cum88_y_2fsk_n=cum88_y_2fsk/y_2fsk_energy^4;
cum88_y_4fsk
=real(cum8x(mom88_y_4fsk,mom44_y_4fsk,mom22_y_4fsk,0,0,0,0,0));
cum88_y_4fsk_n=cum88_y_4fsk/y_4fsk_energy^4;
cum88_y_8fsk
=real(cum8x(mom88_y_8fsk,mom44_y_8fsk,mom22_y_8fsk,0,0,0,0,0));
cum88_y_8fsk_n=cum88_y_8fsk/y_8fsk_energy^4;
cum88_y_2psk
=real(cum8x(mom88_y_2psk,mom44_y_2psk,mom22_y_2psk,0,0,0,0,0));
```

```
cum88_y_2psk_n=cum88_y_2psk/y_2psk_energy^4;
cum88_y_4psk
=real(cum8x(mom88_y_4psk,mom44_y_4psk,mom22_y_4psk,0,0,0,0,0));
cum88_y_4psk_n=cum88_y_4psk/y_4psk_energy^4;
cum88_y_8psk
=real(cum8x(mom88_y_8psk,mom44_y_8psk,mom22_y_8psk,0,0,0,0,0));
cum88_y_8psk_n=cum88_y_8psk/y_8psk_energy^4;
cum88_y_16qam =real(cum8x(mom88_y_16qam ,mom44_y_16qam
,mom22_y_16qam,0,0,0,0,0));
cum88_y_16qam_n=cum88_y_16qam/y_16qam_energy^4;
cum88_y_64qam =real(cum8x(mom88_y_64qam ,mom44_y_64qam
,mom22_y_64qam,0,0,0,0,0));
cum88_y_64qam_n=cum88_y_64qam/y_64qam_energy^4;
cum88_y_256qam =real(cum8x(mom88_y_256qam ,mom44_y_256qam
,mom22_y_256qam,0,0,0,0,0));
cum88_y_256qam_n=cum88_y_256qam/y_256qam_energy^4;
a88=[cum88_y_2fsk_n cum88_y_4fsk_n cum88_y_8fsk_n cum88_y_2psk_n
cum88_y_4psk_n cum88_y_8psk_n cum88_y_16qam_n cum88_y_64qam_n
cum88_y_256qam_n];
SUM_cum88=[SUM_cum88;a88];
a88=[];
% mom44
mom44_y_2fsk_n=mom44_y_2fsk/y_2fsk_energy;
mom44_y_4fsk_n=mom44_y_4fsk/y_4fsk_energy;
mom44_y_8fsk_n=mom44_y_8fsk/y_8fsk_energy;
mom44_y_2psk_n=mom44_y_2psk/y_2psk_energy;
mom44_y_8psk_n=mom44_y_8psk/y_8psk_energy;
mom44_y_16qam_n=mom44_y_16qam/y_16qam_energy;
mom44_y_64qam_n=mom44_y_64qam/y_64qam_energy;
mom44_y_256qam_n=mom44_y_256qam/y_256qam_energy;
am44=[mom44_y_2fsk_n mom44_y_4fsk_n mom44_y_8fsk_n mom44_y_2psk_n
mom44_y_4psk_n mom44_y_8psk_n mom44_y_16qam_n mom44_y_64qam_n
mom44_y_256qam_n];
SUM_mom44=[SUM_mom44;am44];
am44=[];
% mom22
mom22_y_2fsk_n=mom22_y_2fsk/y_2fsk_energy;
mom22_y_4fsk_n=mom22_y_4fsk/y_4fsk_energy;
mom22_y_8fsk_n=mom22_y_8fsk/y_8fsk_energy;
mom22_y_2psk_n=mom22_y_2psk/y_2psk_energy;
mom22_y_4psk_n=mom22_y_4psk/y_4psk_energy;
mom22_y_8psk_n=mom22_y_8psk/y_8psk_energy;
mom22_y_16qam_n=mom22_y_16qam/y_16qam_energy;
mom22_y_64qam_n=mom22_y_64qam/y_64qam_energy;
mom22_y_256qam_n=mom22_y_256qam/y_256qam_energy;
a22=[mom22_y_2fsk_n mom22_y_4fsk_n mom22_y_8fsk_n mom22_y_2psk_n
mom22_y_4psk_n mom22_y_8psk_n mom22_y_16qam_n mom22_y_64qam_n
mom22_y_256qam_n];
SUM_mom22=[SUM_mom22;a22];
a22=[];
% CUM65
mom21_y_2fsk =real(mom2x(y_2fsk,1));
mom21_y_4fsk =real(mom2x(y_4fsk,1));
mom21_y_8fsk =real(mom2x(y_8fsk,1));
```

```
mom21_y_2psk =real(mom2x(y_2psk,1));
mom21_y_4psk =real(mom2x(y_4psk,1));
mom21_y_8psk =real(mom2x(y_8psk,1));
mom21_y_16qam =real(mom2x(y_16qam,1));
mom21_y_64qam =real(mom2x(y_64qam,1));
mom21_y_256qam =real(mom2x(y_256qam,1));
cum65_y_2fsk=(mom65_y_2fsk-10*mom22_y_2fsk*mom43_y_2fsk-
5*mom21_y_2fsk*mom44_y_2fsk+30*(mom22_y_2fsk^2)*mom21_y_2fsk);
cum65_y_2fsk_n=cum65_y_2fsk/(y_2fsk_energy^3)
cum65_y_4fsk=(mom65_y_4fsk-10*mom22_y_4fsk*mom43_y_4fsk-
5*mom21_y_4fsk*mom44_y_4fsk+30*(mom22_y_4fsk^2)*mom21_y_4fsk);
cum65_y_4fsk_n=cum65_y_4fsk/(y_4fsk_energy^3)
cum65_y_8fsk=(mom65_y_8fsk-10*mom22_y_8fsk*mom43_y_8fsk-
5*mom21_y_8fsk*mom44_y_8fsk+30*(mom22_y_8fsk^2)*mom21_y_8fsk);
cum65_y_8fsk_n=cum65_y_8fsk/(y_8fsk_energy^3)
cum65_y_2psk=(mom65_y_2psk-10*mom22_y_2psk*mom43_y_2psk-
5*mom21_y_2psk*mom44_y_2psk+30*(mom22_y_2psk^2)*mom21_y_2psk);
cum65_y_2psk_n=cum65_y_2psk/(y_2psk_energy^3)
cum65_y_4psk=(mom65_y_4psk-10*mom22_y_4psk*mom43_y_4psk-
5*mom21_y_4psk*mom44_y_4psk+30*(mom22_y_4psk^2)*mom21_y_4psk);
cum65_y_4psk_n=cum65_y_4psk/(y_4psk_energy^3)
cum65_y_8psk=(mom65_y_8psk-10*mom22_y_8psk*mom43_y_8psk-
5*mom21_y_8psk*mom44_y_8psk+30*(mom22_y_8psk^2)*mom21_y_8psk);
cum65_y_8psk_n=cum65_y_8psk/(y_8psk_energy^3)
cum65_y_16qam=(mom65_y_16qam-10*mom22_y_16qam*mom43_y_16qam-
5*mom21_y_16qam*mom44_y_16qam+30*(mom22_y_16qam^2)*mom21_y_16qam);
cum65_y_16qam_n=cum65_y_16qam/(y_16qam_energy^3)
cum65_y_64qam=(mom65_y_64qam-10*mom22_y_64qam*mom43_y_64qam-
5*mom21_y_64qam*mom44_y_64qam+30*(mom22_y_64qam^2)*mom21_y_64qam);
cum65_y_64qam_n=cum65_y_64qam/(y_64qam_energy^3)
cum65_y_256qam=(mom65_y_256qam-10*mom22_y_256qam*mom43_y_256qam-
5*mom21_y_256qam*mom44_y_256qam+30*(mom22_y_256qam^2)*mom21_y_256qam);
cum65_y_256qam_n=cum65_y_256qam/(y_256qam_energy^3)
ac65=[cum65_y_2fsk_n cum65_y_4fsk_n cum65_y_8fsk_n cum65_y_2psk_n
cum65_y_4psk_n cum65_y_8psk_n cum65_y_16qam_n cum65_y_64qam_n
cum65_y_256qam_n];
SUM_cum65=[SUM_cum65;ac65];
ac65=[];
% CUM84
cum84_y_2fsk
=real(cum8x(mom84_y_2fsk,mom44_y_2fsk,mom22_y_2fsk,mom84_y_2fsk,mom42_y
_2fsk,mom43_y_2fsk,mom21_y_2fsk,1));
cum84_y_2fsk_n=cum84_y_2fsk/y_2fsk_energy^4;
cum84_y_4fsk
=real(cum8x(mom84_y_4fsk,mom44_y_4fsk,mom22_y_4fsk,mom84_y_4fsk,mom42_y
_4fsk,mom43_y_4fsk,mom21_y_4fsk,1));
cum84_y_4fsk_n=cum84_y_4fsk/y_4fsk_energy^4;
cum84_y_8fsk
=real(cum8x(mom84_y_8fsk,mom44_y_8fsk,mom22_y_8fsk,mom84_y_8fsk,mom42_y
_8fsk,mom43_y_8fsk,mom21_y_8fsk,1));
cum84_y_8fsk_n=cum84_y_8fsk/y_8fsk_energy^4;
cum84_y_2psk
=real(cum8x(mom84_y_2psk,mom44_y_2psk,mom22_y_2psk,mom84_y_2psk,mom42_y
_2psk,mom43_y_2psk,mom21_y_2psk,1));
```

```
cum84_y_2psk_n=cum84_y_2psk/y_2psk_energy^4;
cum84_y_4psk
=real(cum8x(mom84_y_4psk,mom44_y_4psk,mom22_y_4psk,mom84_y_4psk,mom42_y
_4psk,mom43_y_4psk,mom21_y_4psk,1));
cum84_y_4psk_n=cum84_y_4psk/y_4psk_energy^4;
cum84_y_8psk
=real(cum8x(mom84_y_8psk,mom44_y_8psk,mom22_y_8psk,mom84_y_8psk,mom42_y
_8psk,mom43_y_8psk,mom21_y_8psk,1));
cum84_y_8psk_n=cum84_y_8psk/y_8psk_energy^4;
cum84_y_16qam
=real(cum8x(mom84_y_16qam,mom44_y_16qam,mom22_y_16qam,mom84_y_16qam,mom
42_y_16qam,mom43_y_16qam,mom21_y_16qam,1));
cum84_y_16qam_n=cum84_y_16qam/y_16qam_energy^4;
cum84_y_64qam
=real(cum8x(mom84_y_64qam,mom44_y_64qam,mom22_y_64qam,mom84_y_64qam,mom
42_y_64qam,mom43_y_64qam,mom21_y_64qam,1));
cum84_y_64qam_n=cum84_y_64qam/y_64qam_energy^4;
cum84_y_256qam
=real(cum8x(mom84_y_256qam,mom44_y_256qam,mom22_y_256qam,mom84_y_256qam
,mom42_y_256qam,mom43_y_256qam,mom21_y_256qam,1));
cum84_y_256qam_n=cum84_y_256qam/y_256qam_energy^4;
a84=[cum84_y_2fsk_n cum84_y_4fsk_n cum84_y_8fsk_n cum84_y_2psk_n
cum84_y_4psk_n cum84_y_8psk_n cum84_y_16qam_n cum84_y_64qam_n
cum84_y_256qam_n];
SUM_cum84=[SUM_cum84;a84];
a84=[];
end
end
end
% Save all data to the workspace
save classifier_data1 SUM_mom22 SUM_mom43 SUM_mom44 SUM_mom65 SUM_cum44
SUM_cum65 SUM_cum88  SUM_cum84
% Plot all moments and cumulants
figure
plot(1:length(SUM_mom43),SUM_mom43(:,1),'k');hold on
plot(1:length(SUM_mom43),SUM_mom43(:,2),'b');hold on
plot(1:length(SUM_mom43),SUM_mom43(:,3),'g');hold on
plot(1:length(SUM_mom43),SUM_mom43(:,4),'r');hold on
plot(1:length(SUM_mom43),SUM_mom43(:,5),'y');hold on
plot(1:length(SUM_mom43),SUM_mom43(:,6),'m');hold on
plot(1:length(SUM_mom43),SUM_mom43(:,7),'c');hold on
plot(1:length(SUM_mom43),SUM_mom43(:,8),'ob');hold on
plot(1:length(SUM_mom43),SUM_mom43(:,9),'*b')
title('M43')
figure
plot(1:length(SUM_mom65),SUM_mom65(:,1),'k');hold on
plot(1:length(SUM_mom65),SUM_mom65(:,2),'b');hold on
plot(1:length(SUM_mom65),SUM_mom65(:,3),'g');hold on
plot(1:length(SUM_mom65),SUM_mom65(:,4),'r');hold on
plot(1:length(SUM_mom65),SUM_mom65(:,5),'y');hold on
plot(1:length(SUM_mom65),SUM_mom65(:,6),'m');hold on
plot(1:length(SUM_mom65),SUM_mom65(:,7),'c');hold on
plot(1:length(SUM_mom65),SUM_mom65(:,8),'ob');hold on
plot(1:length(SUM_mom65),SUM_mom65(:,9),'*b')
title('M65')
figure
plot(1:length(SUM_mom44),SUM_mom44(:,1),'k');hold on
plot(1:length(SUM_mom44),SUM_mom44(:,2),'b');hold on
plot(1:length(SUM_mom44),SUM_mom44(:,3),'g');hold on
plot(1:length(SUM_mom44),SUM_mom44(:,4),'r');hold on
plot(1:length(SUM_mom44),SUM_mom44(:,5),'y');hold on
plot(1:length(SUM_mom44),SUM_mom44(:,6),'m');hold on
plot(1:length(SUM_mom44),SUM_mom44(:,7),'c');hold on
plot(1:length(SUM_mom44),SUM_mom44(:,8),'ob');hold on
plot(1:length(SUM_mom44),SUM_mom44(:,9),'*b')
title('M44')
figure
plot(1:length(SUM_mom42),SUM_mom42(:,1),'k');hold on
plot(1:length(SUM_mom42),SUM_mom42(:,2),'b');hold on
plot(1:length(SUM_mom42),SUM_mom42(:,3),'g');hold on
plot(1:length(SUM_mom42),SUM_mom42(:,4),'r');hold on
plot(1:length(SUM_mom42),SUM_mom42(:,5),'y');hold on
plot(1:length(SUM_mom42),SUM_mom42(:,6),'m');hold on
plot(1:length(SUM_mom42),SUM_mom42(:,7),'c');hold on
plot(1:length(SUM_mom42),SUM_mom42(:,8),'ob');hold on
plot(1:length(SUM_mom42),SUM_mom42(:,9),'*b')
title('M42')
figure
plot(1:length(SUM_mom84),SUM_mom84(:,1),'k');hold on
plot(1:length(SUM_mom84),SUM_mom84(:,2),'b');hold on
plot(1:length(SUM_mom84),SUM_mom84(:,3),'g');hold on
plot(1:length(SUM_mom84),SUM_mom84(:,4),'r');hold on
plot(1:length(SUM_mom84),SUM_mom84(:,5),'y');hold on
plot(1:length(SUM_mom84),SUM_mom84(:,6),'m');hold on
plot(1:length(SUM_mom84),SUM_mom84(:,7),'c');hold on
plot(1:length(SUM_mom84),SUM_mom84(:,8),'ob');hold on
plot(1:length(SUM_mom84),SUM_mom84(:,9),'*b')
title('M84')
figure
plot(1:length(SUM_mom86),SUM_mom86(:,1),'k');hold on
plot(1:length(SUM_mom86),SUM_mom86(:,2),'b');hold on
plot(1:length(SUM_mom86),SUM_mom86(:,3),'g');hold on
plot(1:length(SUM_mom86),SUM_mom86(:,4),'r');hold on
plot(1:length(SUM_mom86),SUM_mom86(:,5),'y');hold on
plot(1:length(SUM_mom86),SUM_mom86(:,6),'m');hold on
plot(1:length(SUM_mom86),SUM_mom86(:,7),'c');hold on
plot(1:length(SUM_mom86),SUM_mom86(:,8),'ob');hold on
plot(1:length(SUM_mom86),SUM_mom86(:,9),'*b')
title('M86')
figure
plot(1:length(SUM_cum44),SUM_cum44(:,1),'k');hold on
plot(1:length(SUM_cum44),SUM_cum44(:,2),'b');hold on
plot(1:length(SUM_cum44),SUM_cum44(:,3),'g');hold on
plot(1:length(SUM_cum44),SUM_cum44(:,4),'r');hold on
plot(1:length(SUM_cum44),SUM_cum44(:,5),'y');hold on
plot(1:length(SUM_cum44),SUM_cum44(:,6),'m');hold on
plot(1:length(SUM_cum44),SUM_cum44(:,7),'c');hold on
plot(1:length(SUM_cum44),SUM_cum44(:,8),'ob');hold on
plot(1:length(SUM_cum44),SUM_cum44(:,9),'*b')
title('CUM44')
figure
plot(1:length(SUM_cum88),SUM_cum88(:,1),'k');hold on
plot(1:length(SUM_cum88),SUM_cum88(:,2),'b');hold on
plot(1:length(SUM_cum88),SUM_cum88(:,3),'g');hold on
plot(1:length(SUM_cum88),SUM_cum88(:,4),'r');hold on
plot(1:length(SUM_cum88),SUM_cum88(:,5),'y');hold on
plot(1:length(SUM_cum88),SUM_cum88(:,6),'m');hold on
plot(1:length(SUM_cum88),SUM_cum88(:,7),'c');hold on
plot(1:length(SUM_cum88),SUM_cum88(:,8),'ob');hold on
lot(1:length(SUM_cum88),SUM_cum88(:,9),'*b')
title ('CUM88')
figure
plot(1:length(SUM_cum84),SUM_cum84(:,1),'k');hold on
plot(1:length(SUM_cum84),SUM_cum84(:,2),'b');hold on
plot(1:length(SUM_cum84),SUM_cum84(:,3),'g');hold on
plot(1:length(SUM_cum84),SUM_cum84(:,4),'r');hold on
plot(1:length(SUM_cum84),SUM_cum84(:,5),'y');hold on
plot(1:length(SUM_cum84),SUM_cum84(:,6),'m');hold on
plot(1:length(SUM_cum84),SUM_cum84(:,7),'c');hold on
plot(1:length(SUM_cum84),SUM_cum84(:,8),'ob');hold on
plot(1:length(SUM_cum84),SUM_cum84(:,9),'*b')
title ('CUM84')
figure
plot(1:length(SUM_mom22),SUM_mom22(:,1),'k');hold on
plot(1:length(SUM_mom22),SUM_mom22(:,2),'b');hold on
plot(1:length(SUM_mom22),SUM_mom22(:,3),'g');hold on
plot(1:length(SUM_mom22),SUM_mom22(:,4),'r');hold on
plot(1:length(SUM_mom22),SUM_mom22(:,5),'y');hold on
plot(1:length(SUM_mom22),SUM_mom22(:,6),'m');hold on
plot(1:length(SUM_mom22),SUM_mom22(:,7),'c');hold on
plot(1:length(SUM_mom22),SUM_mom22(:,8),'ob');hold on
plot(1:length(SUM_mom22),SUM_mom22(:,9),'*b')
title ('mom22')
figure
plot(1:length(SUM_cum65),SUM_cum65(:,1),'k');hold on
plot(1:length(SUM_cum65),SUM_cum65(:,2),'b');hold on
plot(1:length(SUM_cum65),SUM_cum65(:,3),'g');hold on
plot(1:length(SUM_cum65),SUM_cum65(:,4),'r');hold on
plot(1:length(SUM_cum65),SUM_cum65(:,5),'y');hold on
plot(1:length(SUM_cum65),SUM_cum65(:,6),'m');hold on
plot(1:length(SUM_cum65),SUM_cum65(:,7),'c');hold on
plot(1:length(SUM_cum65),SUM_cum65(:,8),'ob');hold on
plot(1:length(SUM_cum65),SUM_cum65(:,9),'*b')
title ('CUM65')
figure
plot(1:length(SUM_cur),SUM_cur(:,1),'k');hold on
plot(1:length(SUM_cur),SUM_cur(:,2),'b');hold on
plot(1:length(SUM_cur),SUM_cur(:,3),'g');hold on
plot(1:length(SUM_cur),SUM_cur(:,4),'r');hold on
plot(1:length(SUM_cur),SUM_cur(:,5),'y');hold on
plot(1:length(SUM_cur),SUM_cur(:,6),'m');hold on
plot(1:length(SUM_cur),SUM_cur(:,7),'c');hold on
plot(1:length(SUM_cur),SUM_cur(:,8),'ob');hold on
plot(1:length(SUM_cur),SUM_cur(:,9),'*b');
title ('kurtosis')
%                       E N D  O F  F U N C T I O N



function CREATE_SIGNALS(samples)
%**********************************************************************
% Function
%        -Creates the PSK,FSK & QAM signals and stores them in the
workspace
%
% Use: CREATE_SIGNALS(samples)
%
% Input:    samples-> The number desired samples for all modulations
%
% Returns:  The created signals are saved to workspace
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%**********************************************************************
% SIGNAL ATTRIBUTES
fd=1e6;% SET bit rate of signal
N=samples*2 ;% number of samples
samples_to_keep=samples;
bin=256;
snr_db=200;
snr=10^(snr_db/10)
% ***************************************
message=[];
message_mqam=[];
PSK_signals=[];
FSK_signals=[];
QAM_signals=[];
% This loop is for the M of M-QAM for 16,64,256-QAM
   for general_loop1=1 :3
   if general_loop1==1
      Mqam=16;
      M=2;
      bitsqam=4;
      bitsfsk=1;
      bitspsk=1;
   elseif general_loop1==2
      Mqam=64;
      M=4;
      bitsqam=6;
      bitsfsk=2;
      bitspsk=2;
   else
      Mqam=256;
      M=8;
      bitsqam=8;
      bitsfsk=3;
      bitspsk=3;
   end
% I am using 4 samples/symbol
fsymbolqam=fd/bitsqam;
fsymbolfsk=fd/bitsfsk;
fsymbolpsk=fd/bitspsk;
fcarrierqam=fsymbolqam*2;
fcarrierfsk=fsymbolfsk*2;
fcarrierpsk=fsymbolpsk*2;
fsamplingqam=fcarrierqam*2;
fsamplingfsk=fcarrierfsk*2;
fsamplingpsk=fcarrierpsk*2;
npts_qam=fsamplingqam*N/fsymbolqam;
npts_psk=fsamplingpsk*N/fsymbolpsk;
npts_fsk=fsamplingfsk*N/fsymbolfsk;
symbolsqam=N/bitsqam;
symbolspsk=N/bitspsk;
symbolsfsk=N/bitsfsk;
% here is the random message for the psk&fsk
message=randint(1,symbolspsk,M);
%  M-QAM
% will try for 16,64,256 qam
qam_total=[];
%  M-PSK
% will try for 2,4,8 psk
psk_total=[];
%  M-FSK
% will try for 2,4,8 fsk
fsk_total=[];
% here is the random message for the qam
message_mqam=randint(1,symbolsqam,Mqam);
% and here is the mapping for the qam
% We can pick any mapping we want....
mp_qam=modmap(message_mqam,fsymbolqam,fsamplingqam,'qask',Mqam);
% Modulation
 [y_qam2,Y_qam] = getMqam(Mqam,fsamplingqam,fcarrierqam,bin,mp_qam);
 [y_psk2,Y_psk] =
getMpsk(M,fsymbolpsk,fsamplingpsk,fcarrierpsk,bin,message);
 [y_fsk2,Y_fsk] =
getMfsk(M,fsymbolfsk,fsamplingfsk,fcarrierfsk,bin,message);
y_qam=y_qam2(1:samples_to_keep);
y_psk=y_psk2(1:samples_to_keep);
y_fsk=y_fsk2(1:samples_to_keep);
PSK_signals=[PSK_signals y_psk];
FSK_signals=[FSK_signals y_fsk];
QAM_signals=[QAM_signals y_qam];
end
save testing_signals  FSK_signals PSK_signals QAM_signals
%                      E N D  O F  F U N C T I O N
```

```
function net_0=nn_block0(dummy);
%*********************************************************************
% Function
%        - Creates and trains the neural network of block #0
%
% Use: net_0=nn_block0(dummy)
%
% Input:    dummy-> A dummy variable
%
% Returns:  net_0-> The train neural network object of block #0
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*********************************************************************
load classifier_data10
block0_input=[SUM_cum88(:,1);SUM_cum88(:,2);SUM_cum88(:,3);SUM_cum88(:,
4);SUM_cum88(:,5);SUM_cum88(:,6);SUM_cum88(:,7);SUM_cum88(:,8);SUM_cum8
8(:,9)];
k=length(SUM_cum88(:,1));
block0_target=[];
block0_target=ones(3*k,1);
block0_target=[block0_target;-1*ones(k,1)];
block0_target=[block0_target;ones(5*k,1)];
mi=min(min(SUM_cum88));
ma=max(max(SUM_cum88));
net0 = newff([mi ma],[8 1],{'tansig' 'satlins'});
epochs=40;
net0.trainParam.epochs=epochs;
[net_0,tr] = train(net0,block0_input',block0_target');
return
%                      E N D  O F  F U N C T I O N


function net_1=nn_block1(dummy);
%*********************************************************************
% Function
%        - Creates and trains the neural network of block #1
%
% Use: net_1=nn_block1(dummy)
%
% Input:    dummy-> A dummy variable
%
% Returns:  net_1-> The train neural network object of block #1
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*********************************************************************
clear all;load classifier_data10
block1_input_A=[SUM_mom43(:,1);SUM_mom43(:,2);SUM_mom43(:,3);SUM_mom43(
:,4);SUM_mom43(:,5);SUM_mom43(:,6);SUM_mom43(:,7);SUM_mom43(:,8);SUM_mo
m43(:,9)];
```

```
block1_input_B=[SUM_mom22(:,1);SUM_mom22(:,2);SUM_mom22(:,3);SUM_mom22(
:,4);SUM_mom22(:,5);SUM_mom22(:,6);SUM_mom22(:,7);SUM_mom22(:,8);SUM_mo
m22(:,9)];
block1_input=[block1_input_A block1_input_B];
k=length(SUM_mom43(:,1));
block1_target=[];block1_target=ones(3*k,1);
block1_target=[block1_target;-1*ones(k,1)];
block1_target=[block1_target;zeros(5*k,1)];
mi1=min(min(SUM_mom43));ma1=max(max(SUM_mom43));
mi2=min(min(SUM_mom22));ma2=max(max(SUM_mom22));
tmi=[mi1;mi2];tma=[ma1;ma2];
net1 = newff([tmi tma],[20 8 1],{ 'tansig'  'tansig' 'purelin'});
epochs=40;
net1.trainParam.epochs=epochs;
[net_1,tr] = train(net1,block1_input',block1_target');
return
%                        E N D  O F  F U N C T I O N


function net_2=nn_block2(dummy);
%**********************************************************************
% Function
%         - Creates and trains the neural network of block #2
%
% Use: net_2=nn_block2(dummy)
%
% Input:    dummy-> A dummy variable
%
% Returns:  net_2-> The train neural network object of block #2
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%**********************************************************************
load classifier_data10
block2_input=[SUM_cum65(:,1);SUM_cum65(:,2);SUM_cum65(:,3)];
k=length(SUM_cum65(:,1));
block2_target=[];
block2_target=-1*ones(k,1);
block2_target=[block2_target;ones(2*k,1)];
mi=min(min(SUM_cum65));
ma=max(max(SUM_cum65));
net2 = newff([mi ma],[20 10 1],{'tansig' 'tansig'  'satlins'});
epochs=70;
net2.trainParam.epochs=epochs;
[net_2,tr] = train(net2,block2_input',block2_target');
return
%                        E N D  O F  F U N C T I O N



function net_3=nn_block3(dummy);
%**********************************************************************
% Function
%       - Creates and trains the neural network of block #3
%
% Use: net_3=nn_block3(dummy)
%
% Input:    dummy-> A dummy variable
%
% Returns:  net_3-> The train neural network object of block #3
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%**********************************************************************
load classifier_data10
block3_input=[SUM_mom65(:,2);SUM_mom65(:,3)];
k=length(SUM_mom65(:,2));
block3_target=[];
block3_target=ones(k,1);
block3_target=[block3_target;-ones(k,1)];
mi=min(min(SUM_mom65));
ma=max(max(SUM_mom65));
net3 = newff([mi ma],[14 4 2 1],{'tansig' 'tansig' 'tansig'
'purelin'});
epochs=100;
net3.trainParam.epochs=epochs;
[net_3,tr] = train(net3,block3_input',block3_target');
return
%                   E N D   O F   F U N C T I O N


function net_4=nn_block4(dummy);
%**********************************************************************
% Function
%       - Creates and trains the neural network of block #4
%
% Use: net_4=nn_block4(dummy)
%
% Input:    dummy-> A dummy variable
%
% Returns:  net_1-> The train neural network object of block #4
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%**********************************************************************
load classifier_data10
block4_input=[SUM_mom65(:,5);SUM_mom65(:,6);SUM_mom65(:,7);SUM_mom65(:,
8);SUM_mom65(:,9)];
k=length(SUM_mom65(:,6));
block4_target=[];
block4_target=-ones(k,1);
block4_target=[block4_target;ones(k,1)];
block4_target=[block4_target;zeros(3*k,1)];

mi=min(min(SUM_mom65));
ma=max(max(SUM_mom65));
```

```
net4 = newff([mi ma],[20 10 1],{'tansig' 'tansig' 'satlins'});
epochs=40;
net4.trainParam.epochs=epochs;
[net_4,tr] = train(net4,block4_input',block4_target');
return
%                       E N D   O F   F U N C T I O N




function net_5=nn_block5(dummy);
%************************************************************************
% Function
%        - Creates and trains the neural network of block #5
%
% Use: net_5=nn_block5(dummy)
%
% Input:    dummy-> A dummy variable
%
% Returns:  net_5-> The train neural network object of block #5
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%************************************************************************
load classifier_data10
block5_input=[SUM_mom22(:,2);SUM_mom22(:,3)];
k=length(SUM_mom22(:,2));
block5_target=[];
block5_target=-ones(k,1);
block5_target=[block5_target;ones(k,1)];
mi=min(min(SUM_mom22));
ma=max(max(SUM_mom22));
net5 = newff([mi ma],[6 4 1],{'tansig' 'tansig' 'satlins'});
epochs=300;
net5.trainParam.epochs=epochs;
[net_5,tr] = train(net5,block5_input',block5_target');
return
%                       E N D   O F   F U N C T I O N




function net_6=nn_block6(dummy);
%************************************************************************
% Function
%        - Creates and trains the neural network of block #6
%
% Use: net_6=nn_block6(dummy)
%
% Input:    dummy-> A dummy variable
%
```

```
% Returns:  net_6-> The train neural network object of block #6
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*******************************************************************
load classifier_data10
block6_input=[SUM_cum65(:,5);SUM_cum65(:,7);SUM_cum65(:,8);SUM_cum65(:,
9)];
k=length(SUM_cum65(:,1));
block6_target=[];
block6_target=-ones(k,1);
block6_target=[block6_target;ones(3*k,1)];
mi2=min(min(SUM_cum65));
ma2=max(max(SUM_cum65));
net6 = newff([mi2 ma2],[10 1],{'tansig' 'purelin'});
epochs=50;
net6.trainParam.epochs=epochs;
[net_6,tr] = train(net6,block6_input',block6_target');
return
%                       E N D  O F  F U N C T I O N


function train_NNs(dummy);
%*******************************************************************
% Function
%        - Saves into workspace all trained networks from blocks 0 to 4
%
% Use:     train_NNs(dummy)
%
% Input:    dummy-> A dummy variable
%
% Returns:  None
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*******************************************************************
clear all;
% TRAIN ALL BLOCKS
dummy=0;
net_0=nn_block0(dummy);
net_1=nn_block1(dummy);
net_2=nn_block2(dummy);
net_3=nn_block3(dummy);
net_4=nn_block4(dummy);
save trained_NNs net_0 net_1 net_2 net_3 net_4
%                       E N D  O F  F U N C T I O N


function [noisy_signal]=addAWGN(signal,snr)
%*******************************************************************
% Function
%        - Creates a noisy sequence of the desired SNR using additive
white gaussian noise
%
```

```
% Use:      [noisy_signal]=addAWGN(signal,snr)
%
% Input:     signal-> The baseband signal that is desired to be
distorted
%            snr -> The desired signal to noise ratio as a number and
NOT as dB
%
% Returns:  noisy_signal-> The output noisy sequence with the desired
SNR
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*********************************************************************
real_signal=real(signal);
imag_signal=imag(signal);
real_noise=randn(size(real_signal));
imag_noise=randn(size(imag_signal));
real_noise=real_noise-mean(real_noise);
imag_noise=imag_noise-mean(imag_noise);
s1=sum(abs(signal).^2);
sw1=ss1./snr;
sr1=sum(real_noise.^2); si1=sum(imag_noise.^2);
real_noise=real_noise./(sr1.^0.5);imag_noise=imag_noise./(si1.^0.5);
real_noise=real_noise*((sw1./2).^0.5);
imag_noise=imag_noise*((sw1./2).^0.5);
real_signal=real_signal+real_noise;imag_signal=imag_signal+imag_noise;
noise=real_noise+i*imag_noise;
noisy_signal=real_signal+imag_signal.*i;
%                       E N D   O F   F U N C T I O N


function [flag_storage]=ama_function(x_signal);
%*********************************************************************
% Function
%         - Implements the Alphabet Matched Algorithm classifier
%
% Use:     [flag_storage]=ama_function(x_signal)
%
% Input:     signal-> The unknown M-QAM sequence
%
% Returns:  flag_storage-> A flag variable indicating the identified
modulation
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*********************************************************************

flagout=0;
xmax_16qam=0;xmax_64qam=0;xmax_256qam=0;
ymax_16qam=0;ymax_64qam=0;ymax_256qam=0;
centroid_matrix_16qam=[];centroid_matrix_64qam=[];
centroid_matrix_256qam=[];
xvector_16qam=[];xvector_64qam=[];xvector_256qam=[];
yvector_16qam=[];yvector_64qam=[];yvector_256qam=[];
```

```
% Do the FSE-CMA
[x_signal1,flagout]=fsecma(x_signal,0.5);flagout
[x_signal2,flagout]=fsecma(x_signal,5);flagout
if flagout==-1
   disp('we have 16qam')
   x_signal2=x_signal1;
end
[x_signal3,flagout]=fsecma(x_signal,15);flagout
if flagout==-1
   disp('we have 16qam')
   x_signal3=x_signal2;
end
% Do the preprocessing
% Here the signal is processed so that its limits are from -1 to 1
[pn,minp,maxp,tn,mint,maxt] =premnmx(real(x_signal1),imag(x_signal1));
x_signal1=pn+i*tn;
[pn,minp,maxp,tn,mint,maxt] =premnmx(real(x_signal2),imag(x_signal2));
x_signal2=pn+i*tn;
[pn,minp,maxp,tn,mint,maxt] =premnmx(real(x_signal3),imag(x_signal3));
x_signal3=pn+i*tn;
[snr_est1,qam_energy_estimate]=snr_estim(x_signal1);
[snr_est2,qam_energy_estimate]=snr_estim(x_signal2);
[snr_est3,qam_energy_estimate]=snr_estim(x_signal3);
snr_est=mean([snr_est1 snr_est2 snr_est3]);
%***********************************************************************
% The position of the noisy signal's centroids is affected from the
signal to noise ratio
% Therefore an estimate of the SNR helps to fine-tune the theoretical
centroids as close
% to the real centroids as possible.
%***********************************************************************
if snr_est<=8
  xmax_16qam=0.5;
   xmax_64qam=0.4;
   xmax_256qam=0.6;
elseif snr_est>8 & snr_est<=11
   xmax_16qam=0.58;
   xmax_64qam=0.48;
   xmax_256qam=0.7;
elseif snr_est>11 & snr_est<=14
   xmax_16qam=0.6;
   xmax_64qam=0.6;
   xmax_256qam=0.7;
   elseif snr_est>14 & snr_est<=18
   xmax_16qam=0.8;
   xmax_64qam=0.7;

   xmax_256qam=0.7;
elseif snr_est>18
   xmax_16qam=0.8;
   xmax_64qam=0.7;
   xmax_256qam=0.7;
end
ymax_16qam=xmax_16qam;
```

```
ymax_64qam=xmax_64qam;
ymax_256qam=xmax_256qam;
% Create the theoretical centroids of all three M-QAM modulations
xvector_16qam=-xmax_16qam:2*xmax_16qam/3:xmax_16qam;
yvector_16qam=-ymax_16qam:2*ymax_16qam/3:ymax_16qam;
xvector_64qam=-xmax_64qam:2*xmax_64qam/7:xmax_64qam;
yvector_64qam=-ymax_64qam:2*ymax_64qam/7:ymax_64qam;
xvector_256qam=-xmax_256qam:2*xmax_256qam/15:xmax_256qam;
yvector_256qam=-xmax_256qam:2*ymax_256qam/15:ymax_256qam;
for loop1=1:4
  for loop2=1:4
      centroid_matrix_16qam=[centroid_matrix_16qam;xvector_16qam(loop1)
yvector_16qam(loop2)];
   end
end
for loop1=1:8
   for loop2=1:8
      centroid_matrix_64qam=[centroid_matrix_64qam;xvector_64qam(loop1)
yvector_64qam(loop2)];
   end
end
for loop1=1:16
   for loop2=1:16
centroid_matrix_256qam=[centroid_matrix_256qam;xvector_256qam(loop1)
yvector_256qam(loop2)];
   end
end
centroid_vector_16qam=centroid_matrix_16qam(:,1)+i.*centroid_matrix_16q
am(:,2);
centroid_vector_64qam=centroid_matrix_64qam(:,1)+i.*centroid_matrix_64q
am(:,2);
centroid_vector_256qam=centroid_matrix_256qam(:,1)+i.*centroid_matrix_2
56qam(:,2);
%**********************************************************************
% At this point, we have the theoretical centroids and our signal
(already passed from fsecma
% and corrupted with noise)
%**********************************************************************
% I N I T I A L I Z E
% initialize h
samples=length(x_signal1);
taps=20;
g=0.1;
% First filter bank variables declarations
s1=[];CF1=[];
term1_1=[];term2_1=[];

TERM2_1=[];final_1=[];cost_1=0;COST_1=[];COST_function_1=0;TERM3_1=[];h
a_1=[];
h1=zeros(40000,taps);c1=[];
h1(:,taps/2)=1;
c1=centroid_vector_16qam;
M1=16;
sigma1=0.5*(0.2406);
```

```
% Second filter bank  variables declaration
s2=[];CF2=[];
term1_2=[];term2_2=[];
TERM2_2=[];final_2=[];cost_2=0;COST_2=[];COST_function_2=0;TERM3_2=[];h
a_2=[];
h2=zeros(40000,taps);c2=[];
h2(:,taps/2)=1;
c2=centroid_vector_64qam;
M2=64;
sigma2=0.5*(0.1174);
% Third filter bank  variables declaration
s3=[];CF3=[];
term1_3=[];term2_3=[];
TERM2_3=[];final_3=[];cost_3=0;COST_3=[];COST_function_3=0;TERM3_3=[];h
a_3=[];
h3=zeros(40000,taps);c3=[];
h3(:,taps/2)=1;
c3=centroid_vector_256qam;
M3=256;
sigma3=0.5*(0.0584)    ;
flag=2;
% ***  BEGIN AMA  ***
for k=taps:taps:samples-taps-10;
x1=flipud(x_signal1(k:k+taps-1,1));
x2=flipud(x_signal2(k:k+taps-1,1));
x3=flipud(x_signal3(k:k+taps-1,1));
flag=flag+1;
% AMA for first filter bank
for count=1:M1;
term1_1=h1(flag,:)*x1-c1(count);
term2_1=(exp((-
abs(term1_1)^2)/(2*sigma1^2))*((conj(term1_1))/(sigma1^2))*x1)';
TERM2_1=[TERM2_1;(term2_1)];
cost_1=(exp(-(abs(term1_1)^2)/(2*(sigma1^2))));
COST_1=[COST_1;cost_1];
end
COST_1;
TERM3_1=(1/taps)*sum(TERM2_1);
COST_function_1=(1/taps)*(1-(sum(COST_1)));
CF1=[CF1;COST_function_1];
mi1=g*(norm(h1(flag-1,:)*x1)/(norm(TERM3_1)));%((norm(x))^2));%
h1(flag,:)=h1(flag-1,:)-mi1*(TERM3_1);
a1=h1(flag,:)*flipud(x1);
s1=[s1;mi1];TERM3_1=[];TERM2_1=[];term1_1=[];term2_1=[];COST_1=[];
final_1=[final_1;a1];

% AMA for second filter bank

for count=1:M2;
term1_2=h2(flag,:)*x2-c2(count);
term2_2=(exp((-
abs(term1_2)^2)/(2*sigma2^2))*((conj(term1_2))/(sigma2^2))*x2)';
TERM2_2=[TERM2_2;(term2_2)];
cost_2=exp(-(abs(term1_2)^2)/(2*(sigma2^2)));
```

```
COST_2=[COST_2;cost_2];
end
TERM3_2=(1/taps)*sum(TERM2_2);
COST_function_2=(1/taps)*(1-(sum(COST_2)));
CF2=[CF2;COST_function_2];
mi2=g*(norm(h2(flag-1,:)*x2)/(norm(TERM3_2)));
h2(flag,:)=h2(flag-1,:)-mi2*(TERM3_2);
a2=h2(flag,:)*flipud(x2);
s2=[s2;mi2];TERM3_2=[];TERM2_2=[];term1_2=[];term2_2=[];COST_2=[];
final_2=[final_2;a2];
% AMA for third filter bank
for count=1:M3;
term1_3=h3(flag,:)*x3-c3(count);
term2_3=(exp((-
abs(term1_3)^2)/(2*sigma3^2))*((conj(term1_3))/(sigma3^2))*x3)';
TERM2_3=[TERM2_3;(term2_3)];
cost_3=exp(-(abs(term1_3)^2)/(2*(sigma3^2)));
COST_3=[COST_3;cost_3];
end
TERM3_3=(1/taps)*sum(TERM2_3);
COST_function_3=(1/taps)*(1-(sum(COST_3)));
CF3=[CF3;COST_function_3];
mi3=g*(norm(h3(flag-1,:)*x3)/(norm(TERM3_3)));
h3(flag,:)=h3(flag-1,:)-mi3*(TERM3_3);
a3=h3(flag,:)*flipud(x3);
s3=[s3;mi3];
TERM3_3=[];TERM2_3=[];term1_3=[];term2_3=[];COST_3=[];
final_3=[final_3;a3];
end
d1=hist(CF1,16);d2=hist(CF2,64);d3=hist(CF3,256);
criterion=[sum(d1(1:4)) sum(d2(1:16)) sum(d3(1:64))];
i=find(criterion==max(criterion));
if i==1
  disp('we have 16QAM');flag=70;flag_storage=7;
elseif i==2
   disp('we have 64QAM');flag=80;flag_storage=8;
elseif i==3
   disp('we have 256QAM');flag=90;flag_storage=9;
elseif  i(1)==2
   disp ('we have 64QAM or 256QAM');flag_storage=100;
elseif i(1)==1
   disp('we have 16QAM or 64QAM');flag_storage=101;
else
   disp('we have 16QAM or 256QAM');flag_storage=102;
end


return
%                       E N D  O F  F U N C T I O N


function cumulant8= cum8x(m88,m44,m22,m84,m42,m43,m21,flag)
%************************************************************************
```

```
% Function
%        - Calclulates the eighth order cumulants of a sequence
%
% Use:     cumulant8= cum8x(m88,m44,m22,m84,m42,m43,m21,flag)
%
% Input:     m88-> The eighth order moment (M88) of the signal
%            m44-> The fourth order moment (M44) of the signal
%            m22-> The second order moment (M22) of the signal
%            m84-> The eighth order moment (M84) of the signal
%            m42-> The fourth order moment (M42) of the signal
%            m43-> The fourth order moment (M43) of the signal
%            m21-> The second order moment (M42) of the signal
%            flag-> A flag variable indication which cumulant is to be
estimated (C84 or C88)
%
% Returns:  cumulant8-> The eighth order cumulant estimate
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*******************************************************************
if flag==1
   cumulant8=m84-m44.^2-18*m42^2-16*m43.^2-54*m22.^4-144*m21.^4-
432*m22.^2.*m21.^2+12*m44.*m22.^2+96*m43.*m21.*m22+144*m42.*m21.^2+72*m
42.*m22.^2+96*m43.*m22.*m21; %C84
elseif flag==0
   cumulant8=m88-35*m44.^2-630*m22.^4+420*(m22.^2).*m44;% C88
end
return
%                      E N D  O F  F U N C T I O N


function [ynew,flagout]=fsecma(r,stp);
%*******************************************************************
% Function
%        - Implements the FSE-CMA blind equalization algorithm
%
% Use:     [ynew,flagout]=fsecma(r,stp)
%
% Input:     r-> The signal that is to be equalized
%            stp-> The desired algorithm step
%
% Returns:  ynew-> The equalized signal
%            flagout-> A diagnostic flag variable
%
% Function fsecma.m created by the MPRG group
% Modified on 21 January 2001 by
% LtJg George Hatzichristos Hellenic Navy
%*******************************************************************
% Run CMA on T/2-spaced modem data
% with a T/2-spaced equalizer (FSE)
flagout=1;
r=r';
% Get number of T-spaced symbols
```

```
L=(length(r)/2);
%  Normalize to unit power
r=r-mean(r);
r=r/((1/length(r))*norm(r,2)^2);
% Define FSE
Nf=16;  % This is the number of coefficients in use
f=zeros(Nf,L);
% Center spike init
f(Nf/2,Nf/2-1)=1;
% Define step-size & dispersion constant
% any number for g will work to open eye
% or rings
mu=stp;
qam1=abs(r).^4;qam2=abs(r).^2;g=qam1/qam2;
% Define error and equalizer output
e=zeros(1,L);
y=zeros(1,L);
% Run CMA
for k=Nf:2:2*L,
j=k/2;
R=r(k:-1:k-Nf+1).';
y(j)=R.'*f(:,j-1);
if  norm(y(j))>10000
 flagout=-1
    return
end
    f(:,j)=f(:,j-1)+mu*conj(R)*y(j)*(g-abs(y(j))^2);
end
% Run new data to get eye diagram
% make sure to get odd samples
ynew=filter(f(:,j),1,r);
ynew=ynew(2:2:length(ynew));
ynew=ynew(100:length(ynew)-100);
ynew=ynew';
flagout=1;
return
%                       E N D   O F   F U N C T I O N


function [y_fsk,Y_fsk] = getMfsk (M,fd,fs,fc,bin,message);
%**********************************************************************
% Function
%        - Creates an M-FSK signal
%
% Use:     [y_fsk,Y_fsk] = getMfsk (M,fd,fs,fc,bin,message)
%
% Input:     M-> The constellation order (ex. M=8 for 8-FSK)
%            fd-> The digital signal's frequency before the modulation
%            fs-> The sampling frequency
%            fc-> The carrier frequency (For pass-band signals only)
%            bin-> The number of fft bins
%            message->The message that is to be modulated
%
```

```
% Returns:  y_fsk-> The modulated M-fsk sequence
%           Y_fsk-> The spectrum of the M-fsk sequence
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*********************************************************************
% Map the message
mp_fsk=modmap(message,fd,fs,'fsk',M);
% Modulation
%y_fsk=dmod(mp_fsk,fc,fd,fs,'fsk/nomap',M);% PASSBAND SIM
y_fsk=dmodce(mp_fsk,fd,fs,['fsk','/nomap'],M);% BASEBAND SIM
Y_fsk=(abs(fft(y_fsk,bin)));
return
%                       E N D  O F  F U N C T I O N
```

```
function [y_psk,Y_psk] = getMfsk (M,fd,fs,fc,bin,message);
%*********************************************************************
% Function
%        - Creates an M-PSK signal
%
% Use:     [y_psk,Y_psk] = getMpsk (M,fd,fs,fc,bin,message)
%
% Input:     M-> The constellation order (ex. M=8 for 8-PSK)
%            fd-> The digital signal's frequency before the modulation
%            fs-> The sampling frequency
%            fc-> The carrier frequency (For pass-band signals only)
%            bin-> The number of fft bins
%            message->The message that is to be modulated
%
% Returns:  y_psk-> The modulated M-psk sequence
%           Y_psk-> The spectrum of the M-psk sequence
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*********************************************************************
% Map the message
mp_psk=modmap(message,fd,fs,'psk',M);
% Modulation
%y_fsk=dmod(mp_fsk,fc,fd,fs,'fsk/nomap',M);% PASSBAND SIM
y_psk=dmodce(mp_psk,fd,fs,['psk','/nomap'],M);% BASEBAND SIM
Y_psk=(abs(fft(y_psk,bin)));
return
%                       E N D  O F  F U N C T I O N

function [y_qam,Y_qam] = getMqam (Mqam,fs,fc,bin,mp_qam);
%*********************************************************************
```

```
% Function
%        - Creates an M-QAM signal
%
% Use:     [y_qam,Y_qam] = getMqam (Mqam,fs,fc,bin,mp_qam)
%
% Input:     Mqam-> The constellation order (ex. M=16 for 16-QAM)
%            fs-> The sampling frequency
%            fc-> The carrier frequency (For pass-band signals only)
%            bin-> The number of fft bins
%            mp_qam->The message that is to be modulated
%
% Returns:  y_qam-> The modulated M-qam sequence
%           Y_qam-> The spectrum of the M-qam sequence
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*******************************************************************
%y_qam=amod(mp_qam,fc,fs,'qam'); % This is for pass band simulation
y_qam=amodce(mp_qam,fs,'qam');
Y_qam=(abs(fft(y_qam,bin)));
return
%                       E N D   O F   F U N C T I O N


function [y_qam,Y_qam] = getMqam (Mqam,fs,fc,bin,mp_qam);
%*******************************************************************
% Function
%        - Creates an M-QAM signal
%
% Use:     [y_qam,Y_qam] = getMqam (Mqam,fs,fc,bin,mp_qam)
%
% Input:     Mqam-> The constellation order (ex. M=16 for 16-QAM)
%            fs-> The sampling frequency
%            fc-> The carrier frequency (For pass-band signals only)
%            bin-> The number of fft bins
%            mp_qam->The message that is to be modulated
%
% Returns:  y_qam-> The modulated M-qam sequence
%           Y_qam-> The spectrum of the M-qam sequence
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*******************************************************************
%y_qam=amod(mp_qam,fc,fs,'qam'); % This is for pass band simulation
y_qam=amodce(mp_qam,fs,'qam');
Y_qam=(abs(fft(y_qam,bin)));
return
%                       E N D   O F   F U N C T I O N

function moment4= mom4x(signal,flag)
%*******************************************************************
% Function
%        - Calclulates the fourth order moments of a sequence
%
% Use:      moment4= mom4x(signal,flag)
%
% Input:    signal-> The sequence whose moment is to be estimated
%           flag-> A flag variable indication which moment is to be
estimated (M44,M43 or M42)
%
% Returns:  moment4-> The fourth order moment estimate
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*********************************************************************
a=real(signal);
b=imag(signal);
if flag==2
   moment4=(mean(a.^4+b.^4+2*(a.^2).*(b.^2))); % E42
elseif flag==1
    moment4=(mean(a.^4-b.^4)); %E43
elseif flag==0
   moment4=(mean(a.^4+b.^4-6*(a.^2).*(b.^2)));% E44
end
a=[];b=[];
return
%                      E N D  O F  F U N C T I O N


function moment6= mom6x(signal)
%*********************************************************************
% Function
%        - Calclulates the sixth order moment E65 of a sequence
%
% Use:      moment6= mom6x(signal)
%
% Input:    signal-> The sequence whose moment is to be estimated
%
% Returns:  moment6-> The sixth order moment estimate
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%*********************************************************************
a=real(signal);
b=imag(signal);
moment6=(mean(a.^6+b.^6-5*(a.^2).*(b.^4)-5*(a.^4).*(b.^2))); %E65
a=[];b=[];
return
%                      E N D  O F  F U N C T I O N

function moment8= mom8x(signal,flag)
%*********************************************************************
% Function
%        - Calclulates the eighth order moments of a sequence
%
% Use:       moment8= mom8x(signal,flag)
%
% Input:     signal-> The sequence whose moment is to be estimated
%            flag-> A flag variable indication which moment is to be
estimated (M84,M86 or M88)
%
% Returns:  moment8-> The eighth order moment estimate
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%**********************************************************************
a=real(signal);
b=imag(signal);
if flag==1
moment8=(mean(a.^8+b.^8+4*(a.^6.*b.^2)+6*(a.^4.*b.^4)+4*(a.^2.*b.^6)));
%E84
elseif flag==0
   moment8=(mean(a.^8+b.^8-28*(a.^6.*b.^2)+70*(a.^4.*b.^4)-
28*(a.^2.*b.^6)));% E88
elseif flag==2
   moment8=mean(a.^8+b.^8-4*(a.^6).*(b.^2)-10.*(a.^4).*(b.^4)-
4.*(a.^2).*(b.^6));% E86
end
a=[];b=[];
return
%                        E N D  O F  F U N C T I O N



function [snr_es_db,qam_energy_estimate]=snr_estim(noisy_qam);
%**********************************************************************
% Function
%         - Estimates the signal to noise ratio of a M-QAM sequence
using the kurtosis
%           of the noisy signal, since the kurtosis is proportional to
the noise level
%           and is equal to 3 for a 0dB SNR.
%
% Use:       [snr_es_db,qam_energy_estimate]=snr_estim(noisy_qam)
%
% Input:     noisy_qam-> The sequence whose SNR is to be estimated
%
% Returns:  snr_es_db-> The signal to noise ratio estimate
%           qam_energy_estimate-> The estimated energy of the noiseless
M-qam signal
%
% 21 January 2001
% LtJg George Hatzichristos Hellenic Navy
%**********************************************************************
noisy_qam_energy=(1/length(noisy_qam))*norm(noisy_qam,2)^2
m2_sn=kurtosis(real(noisy_qam));
```

```
table_qam_snr=[];table_estim_qam=[];difference=[];table_qam=[];
table_qam=[20      17    14     10     7       4     1      0;
   1.8016 1.826 1.8675 1.9913 2.1507 2.3728 2.6188 2.693];
table_qam_snr=table_qam(2,:);
table_estim_qam=ones(1,8);
table_estim_qam=m2_sn*table_estim_qam;
difference=abs(table_qam_snr-table_estim_qam);
i=find(difference==min(difference));
snr_es_db=0;snr_es_db=table_qam(1,i);
snr_es=10^(snr_es_db/10);noisy_qam_energy;
qam_energy_estimate=(snr_es*noisy_qam_energy)/(1+snr_es);
return
%                      E N D  O F  F U N C T I O N
```

APPENDIX E. HIGHER ORDER STATISTICS BEHAVIOR IN NOISE AND FADING MULTIPATH ENVIRONMENTS

The robustness of higher order statistics in noise and propagation phenomena is a key to the success of the proposed classifier. Marchand [MAR98] recommends the use of moments and cumulants for the classification of digital modulations but does not present any clues about the robustness of these tools in real world situations. These situations are simulated and presented next.The simulation results are divided into two categories. In the first category only the additive white gaussian noise channel is considered. In the second category, nine different propagation channels (Appendix C, Figures C-1 to C-9) are used in addition to white noise. Each category includes three different sets of results. 1000, 15000 and 30000 signal samples respectively, are used to indicate the minimum required samples for clear separation between all features.

E.1 Additive White Gaussian Noice channel simulations

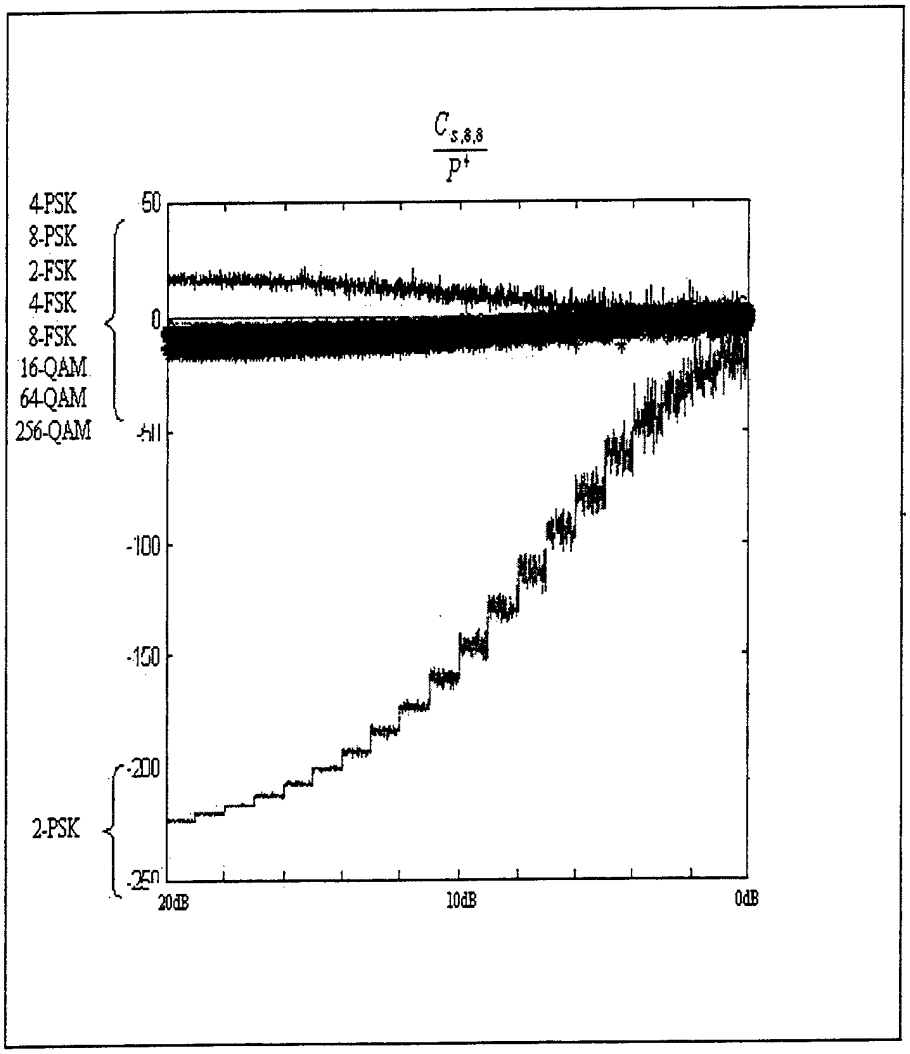


Figure E1-1. $\frac{C_{s,8,8}}{P^4}$, 1000 samples data-set, 100 trials per SNR level.

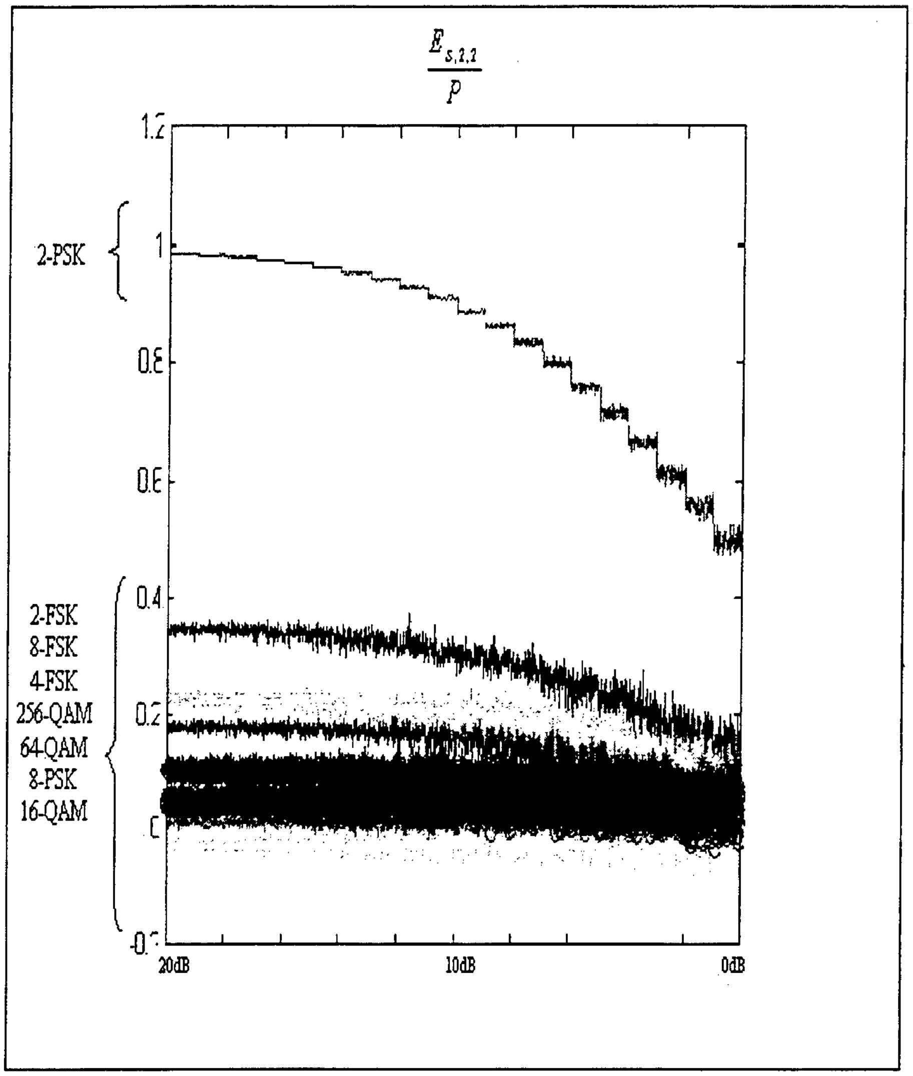


Figure E1-2. $\frac{E_{S,2,2}}{P}$, 1000 samples data-set, 100 trials per SNR level.

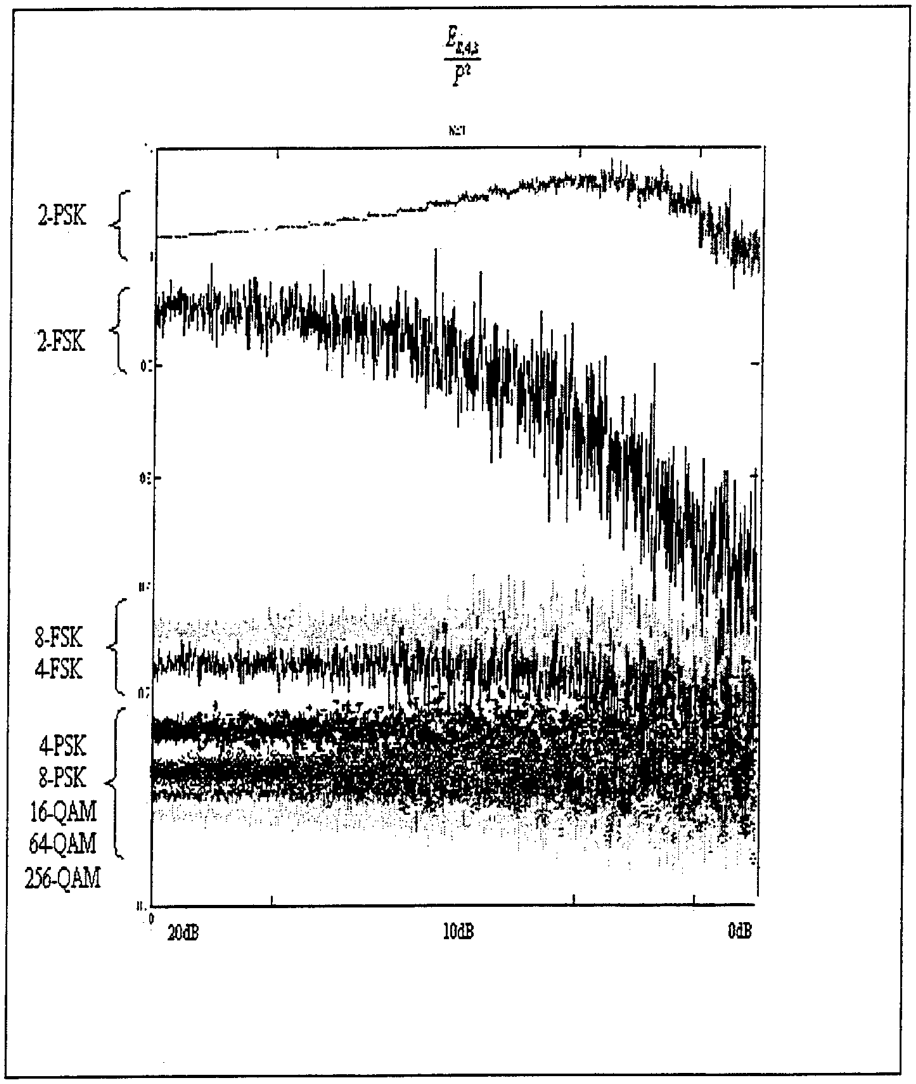


Figure E1-3. $\frac{E_{S,4,3}}{P^2}$, 1000 samples data-set, 100 trials per SNR level.

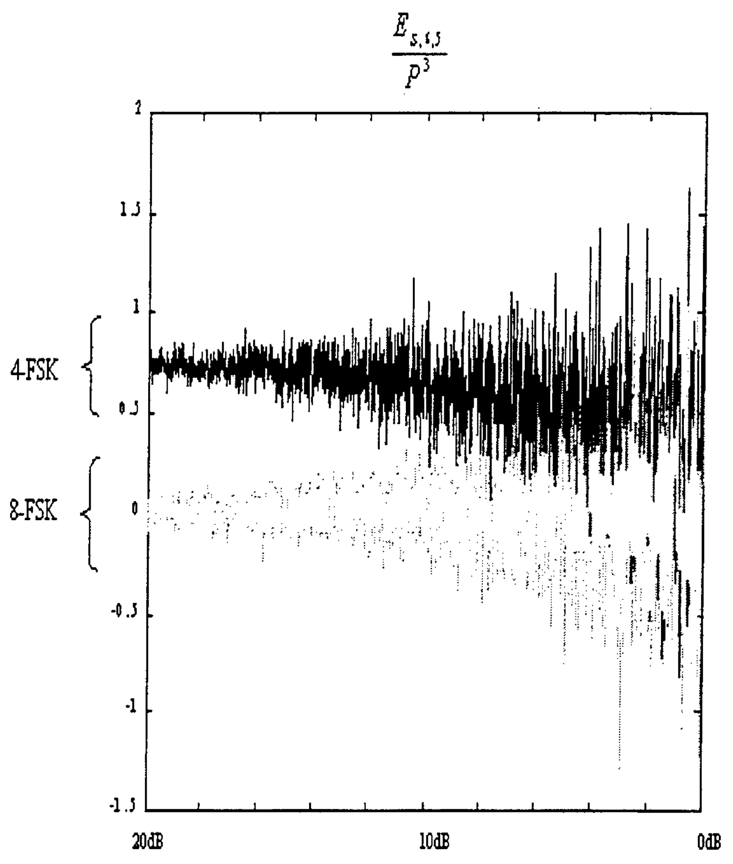


Figure E1-4. $\frac{E_{S,6,5}}{P^3}$, 1000 samples data-set, 100 trials per SNR level.

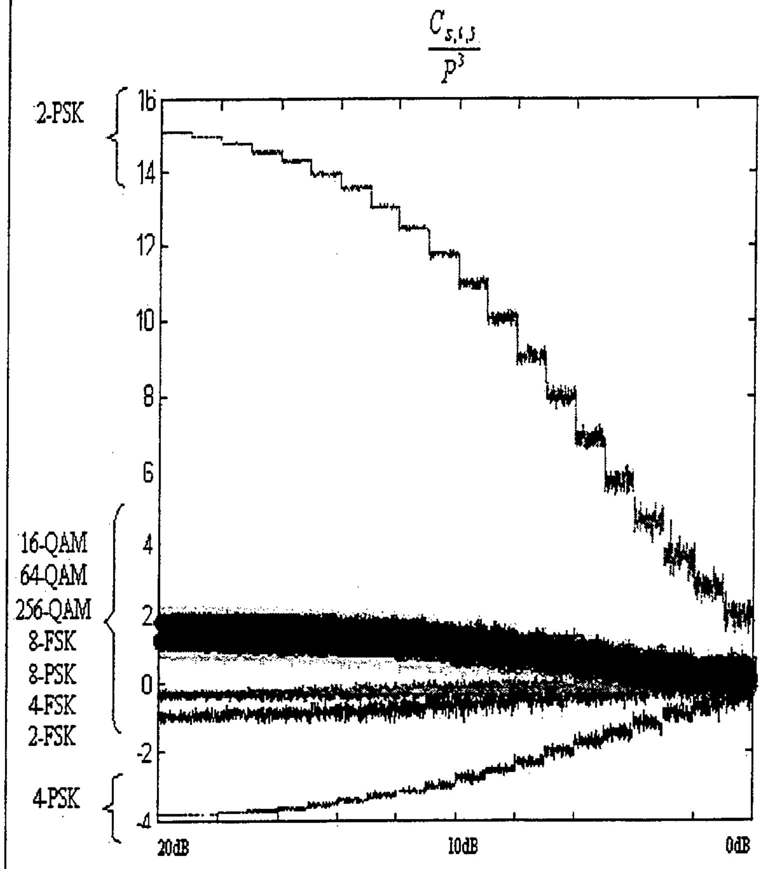


Figure E1-5. $\frac{C_{s,6,5}}{P^3}$, 1000 samples data-set, 100 trials per SNR level.

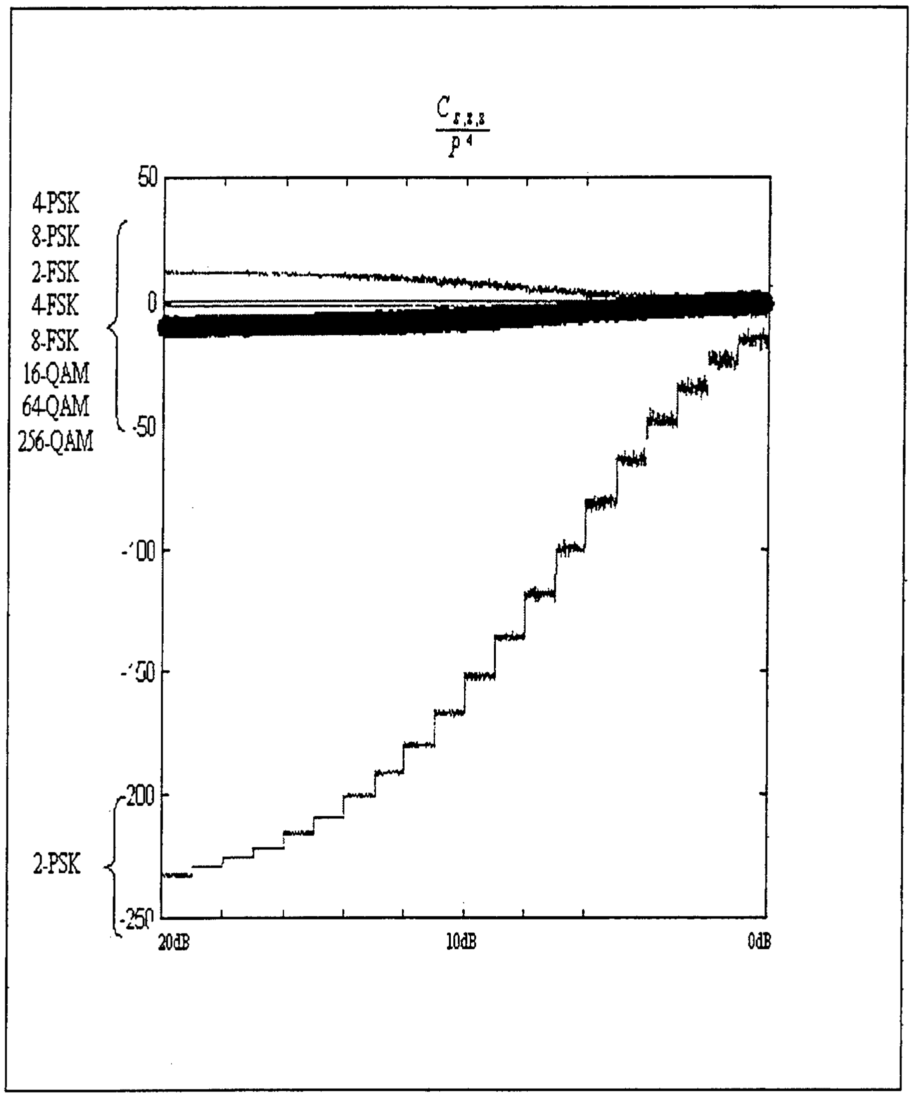


Figure E1-6. $\frac{C_{S,8,8}}{P^4}$, 15,000 samples data-set, 100 trials per SNR level.

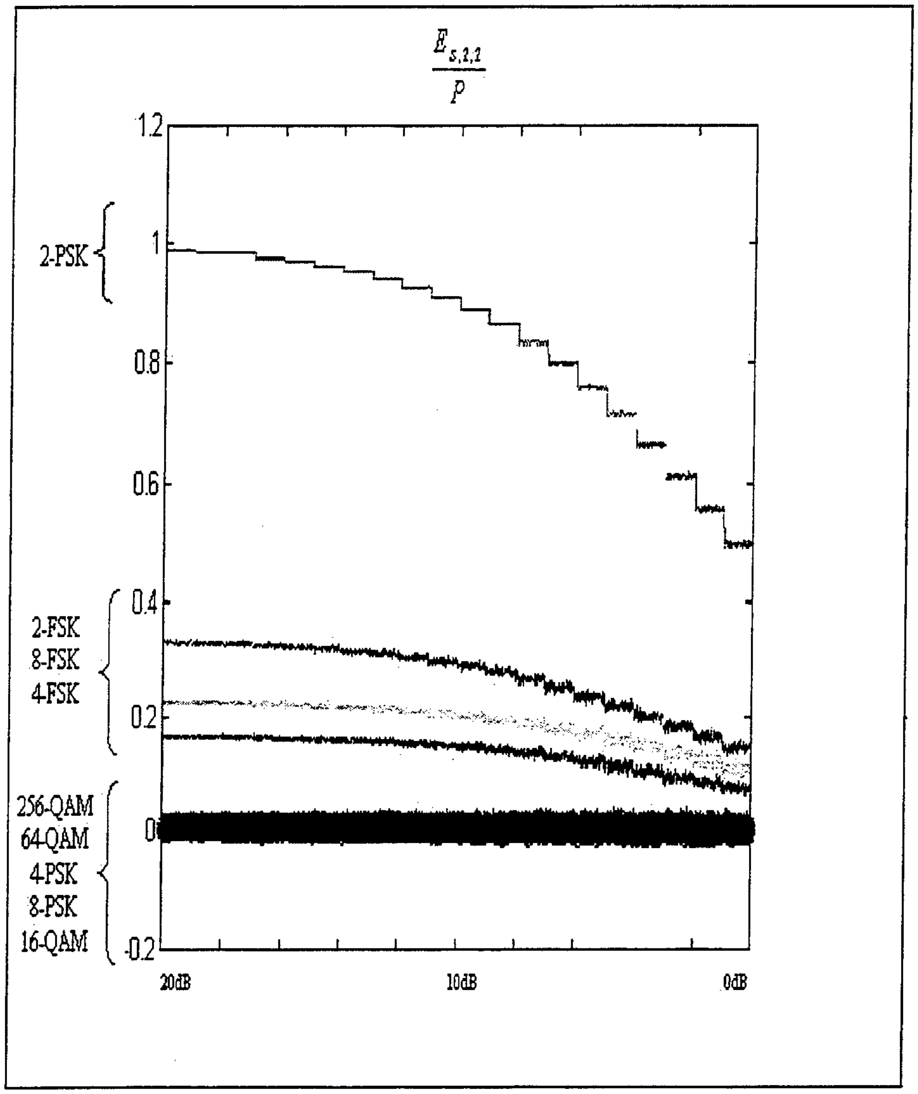


Figure E1-7. $\frac{E_{s,2,2}}{P}$, 15,000 samples data-set, 100 trials per SNR level.

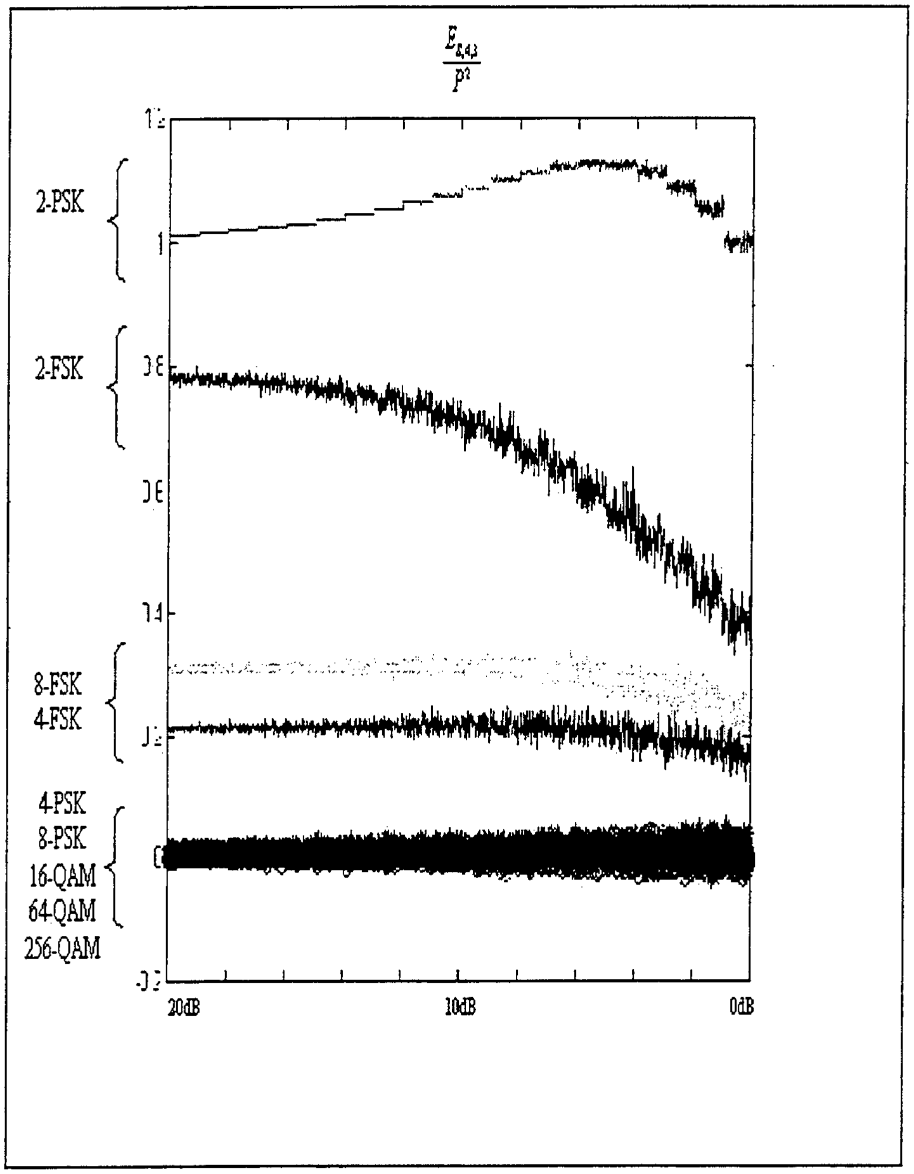


Figure E1-8. $\frac{E_{S,4,3}}{P^2}$, 15,000 samples data-set, 100 trials per SNR level.

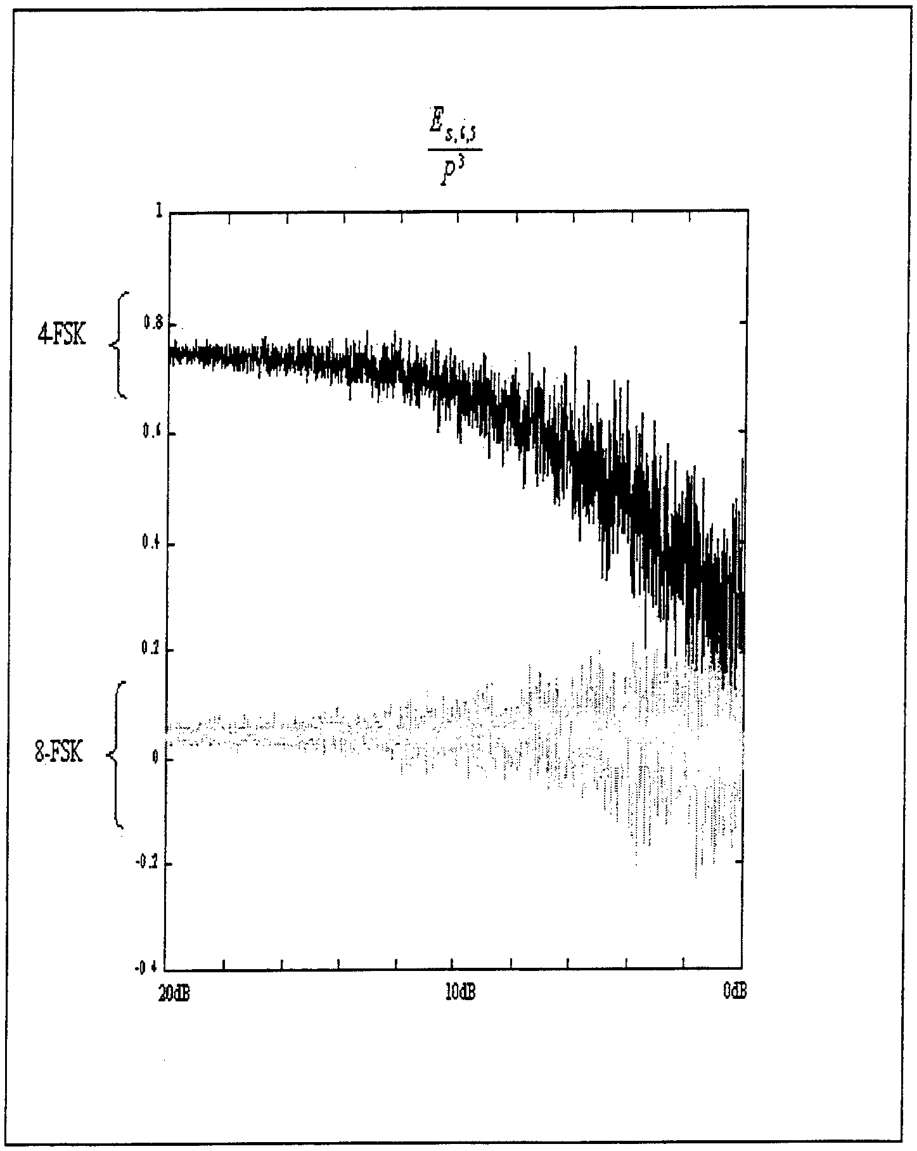


Figure E1-9. $\frac{E_{S,6,5}}{P^3}$, 15,000 samples data-set, 100 trials per SNR level.

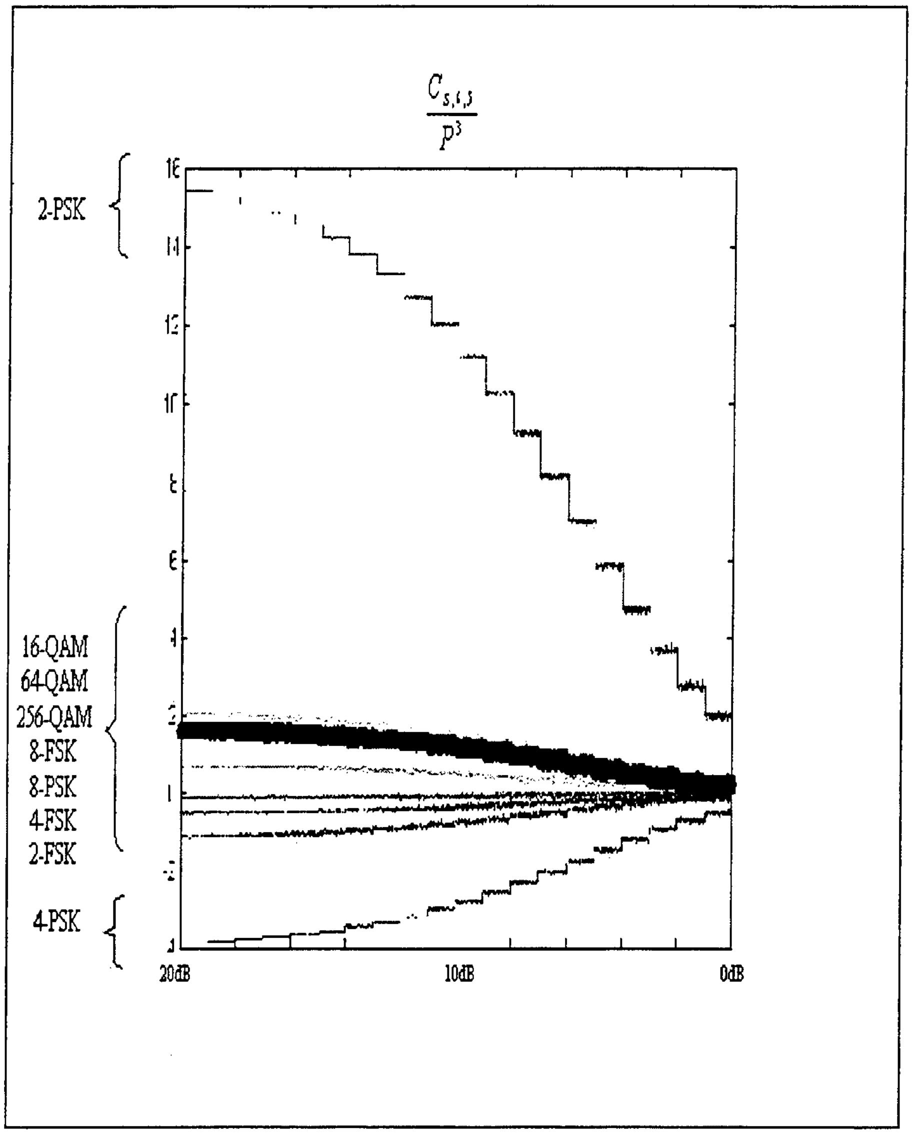


Figure E1-10. $\frac{C_{S,6,5}}{P^3}$, 15,000 samples data-set, 100 trials per SNR level.

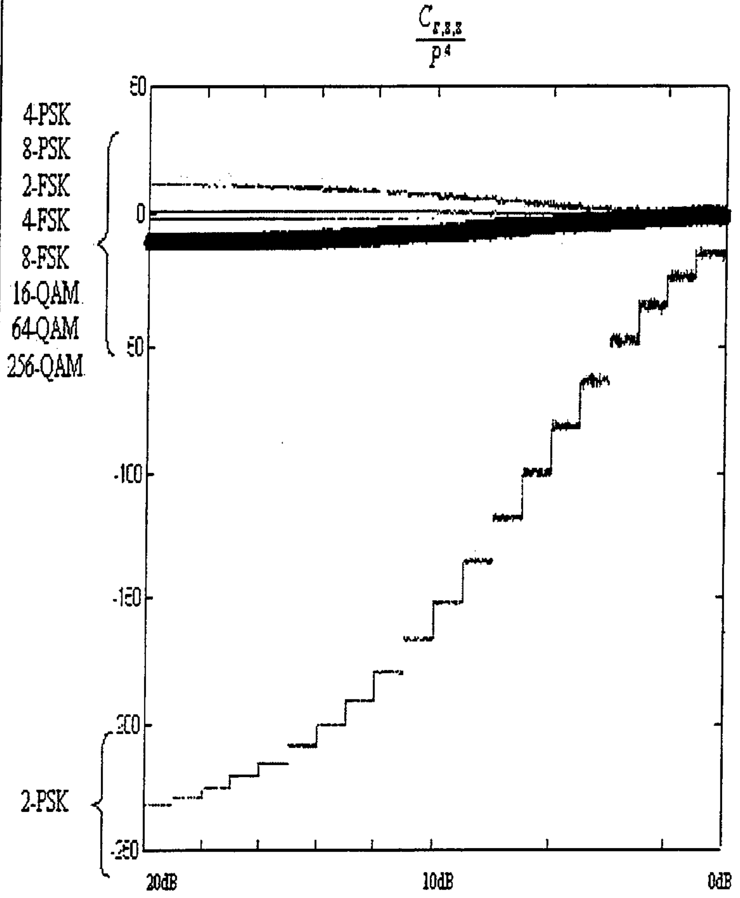


Figure E1-11. $\frac{C_{S,8,8}}{P^4}$, 30,000 samples data-set, 100 trials per SNR level.

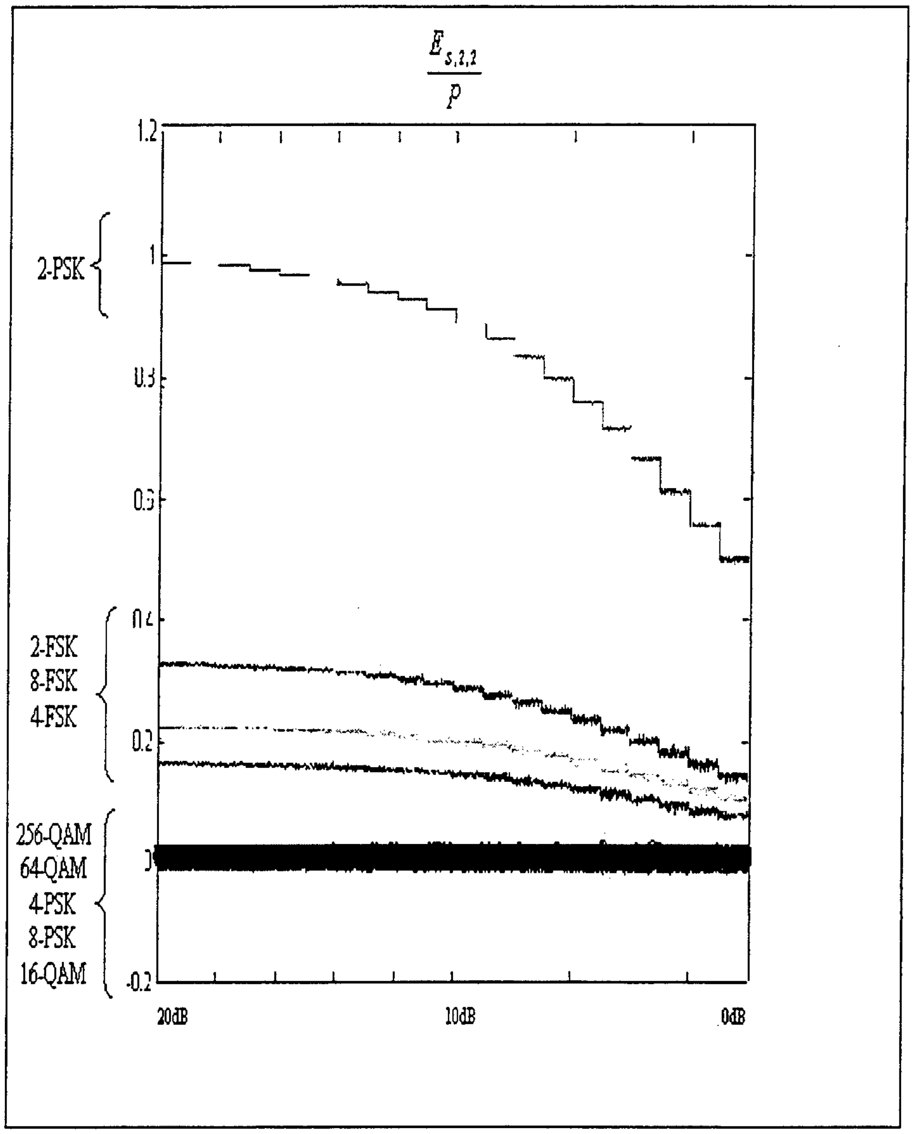


Figure E1-12. $\frac{E_{s,2,2}}{P}$, 30,000 samples data-set, 100 trials per SNR level.

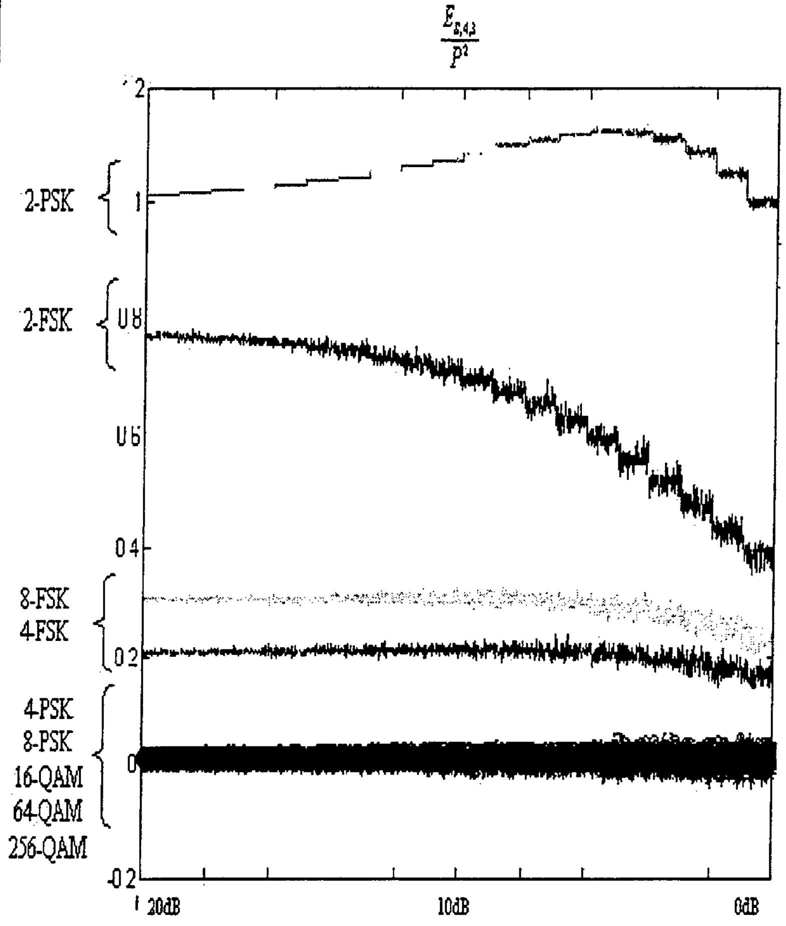


Figure E1-13. $\frac{E_{S,4,3}}{P^2}$, 30,000 samples data-set, 100 trials per SNR level.

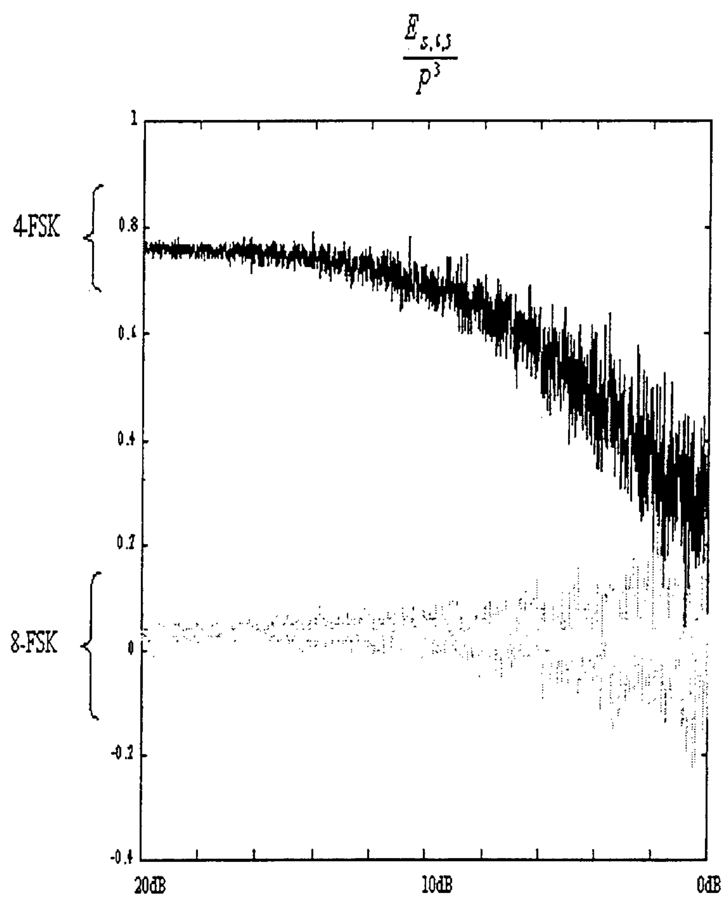


Figure E1-14. $\frac{E_{S,6,5}}{P^3}$, 30,000 samples data-set, 100 trials per SNR level.

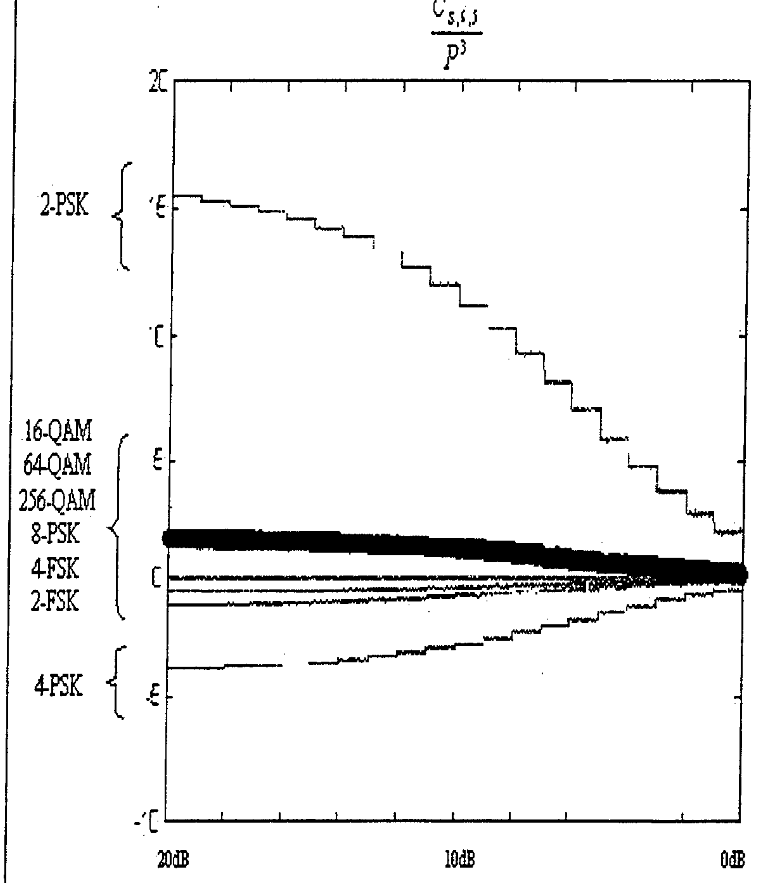


Figure E1-15. $\frac{C_{S,6,5}}{P^3}$, 30,000 samples data-set, 100 trials per SNR level.

E.2 Fading multi-path channels simulations

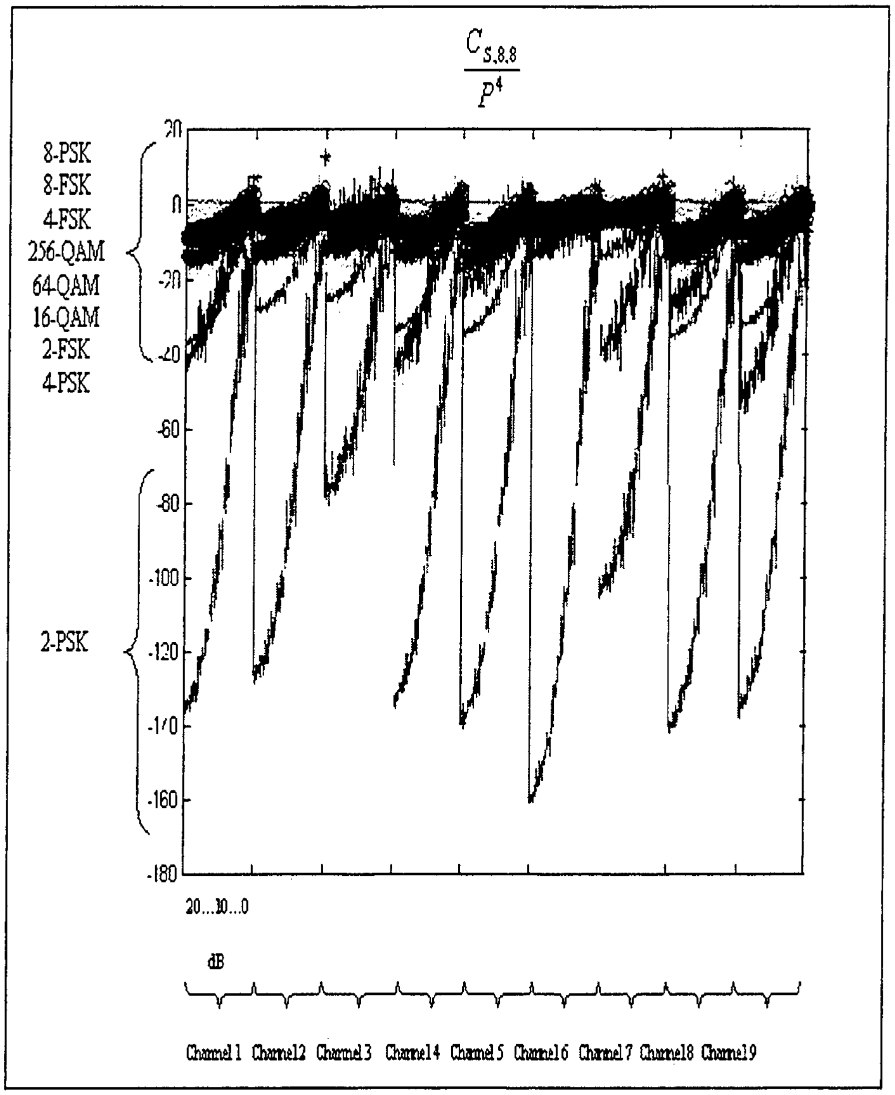


Figure E2-1. $\frac{C_{S,8,8}}{P^4}$, 1000 samples data-set, 100 trials per SNR level.

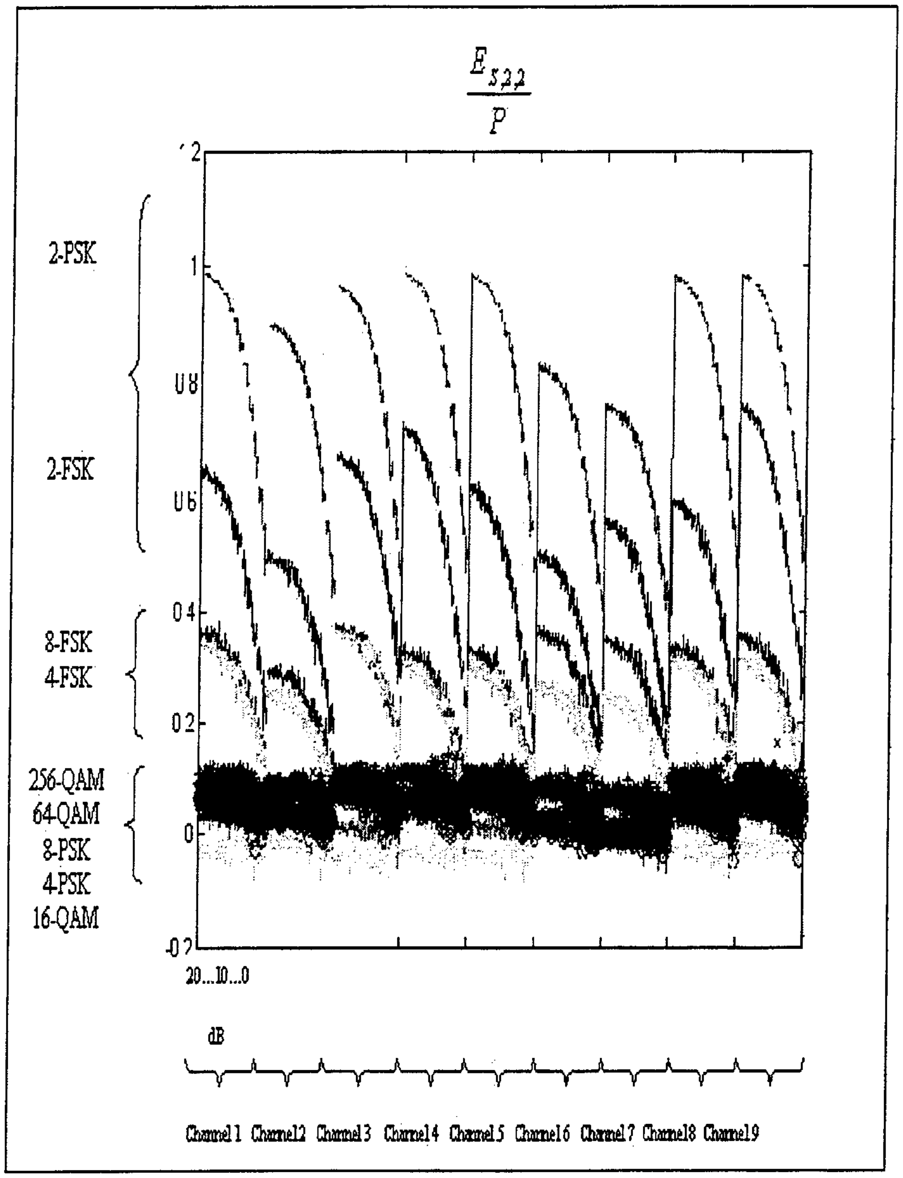


Figure E2-2. $\frac{E_{S,2,2}}{P}$, 1000 samples data-set, 100 trials per SNR level.

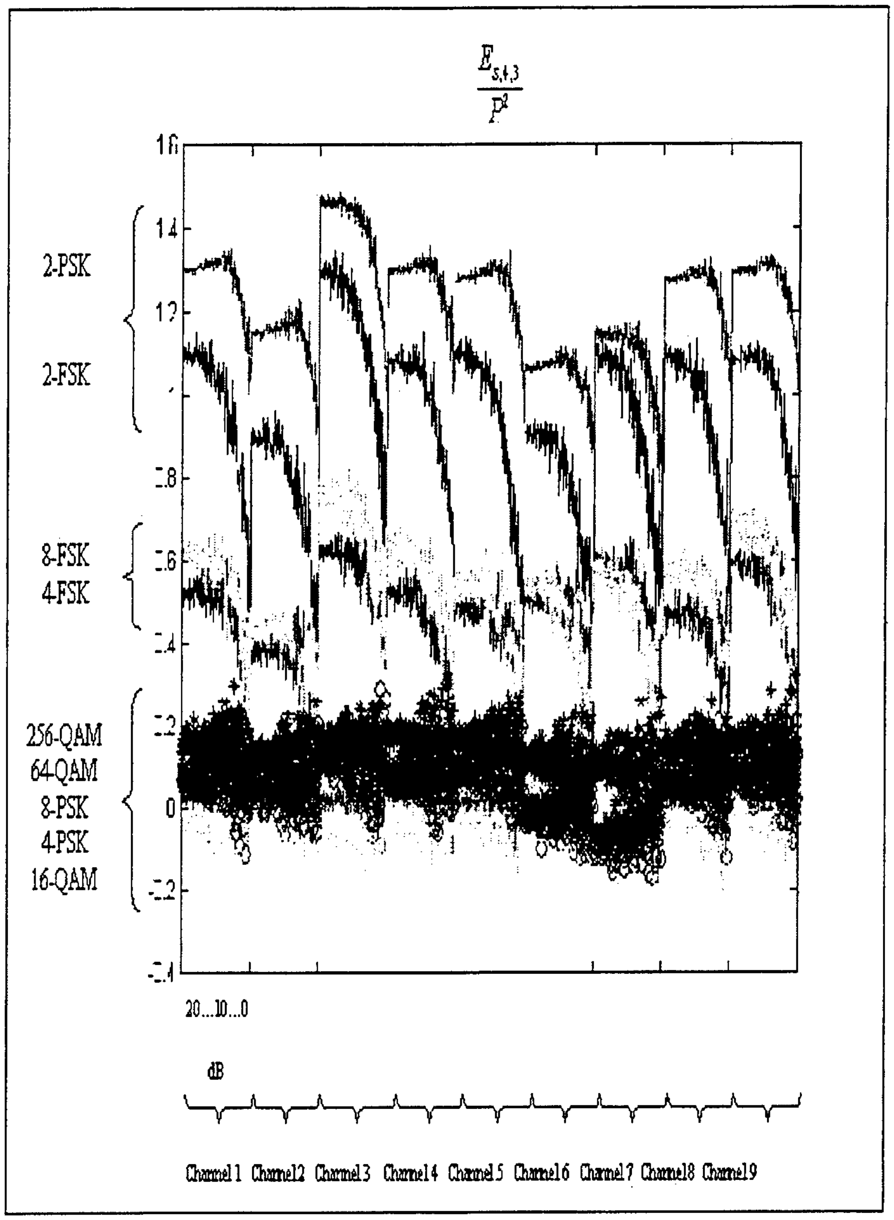


Figure E2-3. $\frac{E_{S,4,3}}{P^2}$, 1000 samples data-set, 100 trials per SNR level.

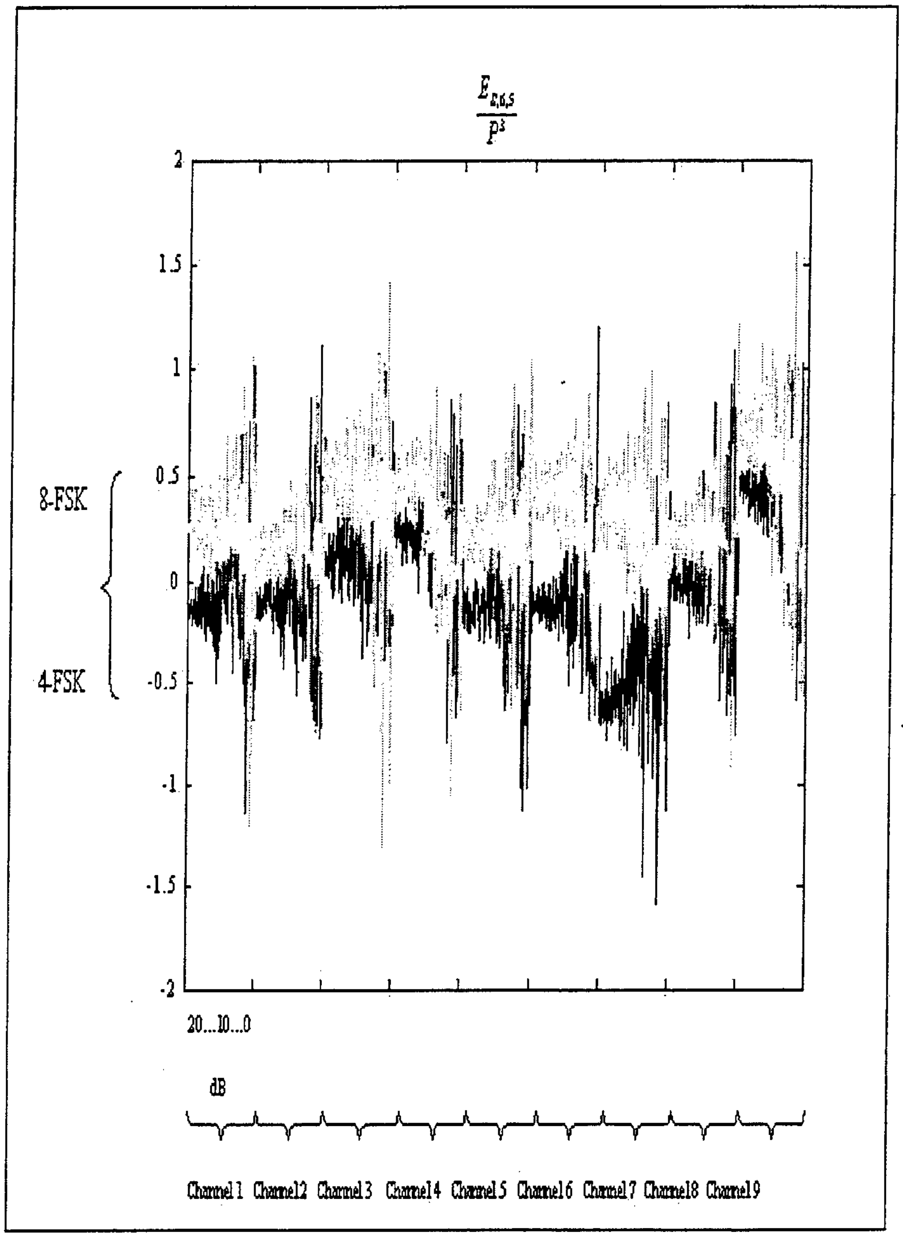


Figure E2-4. $\frac{E_{S.6.5}}{P^3}$, 1000 samples data-set, 100 trials per SNR level.

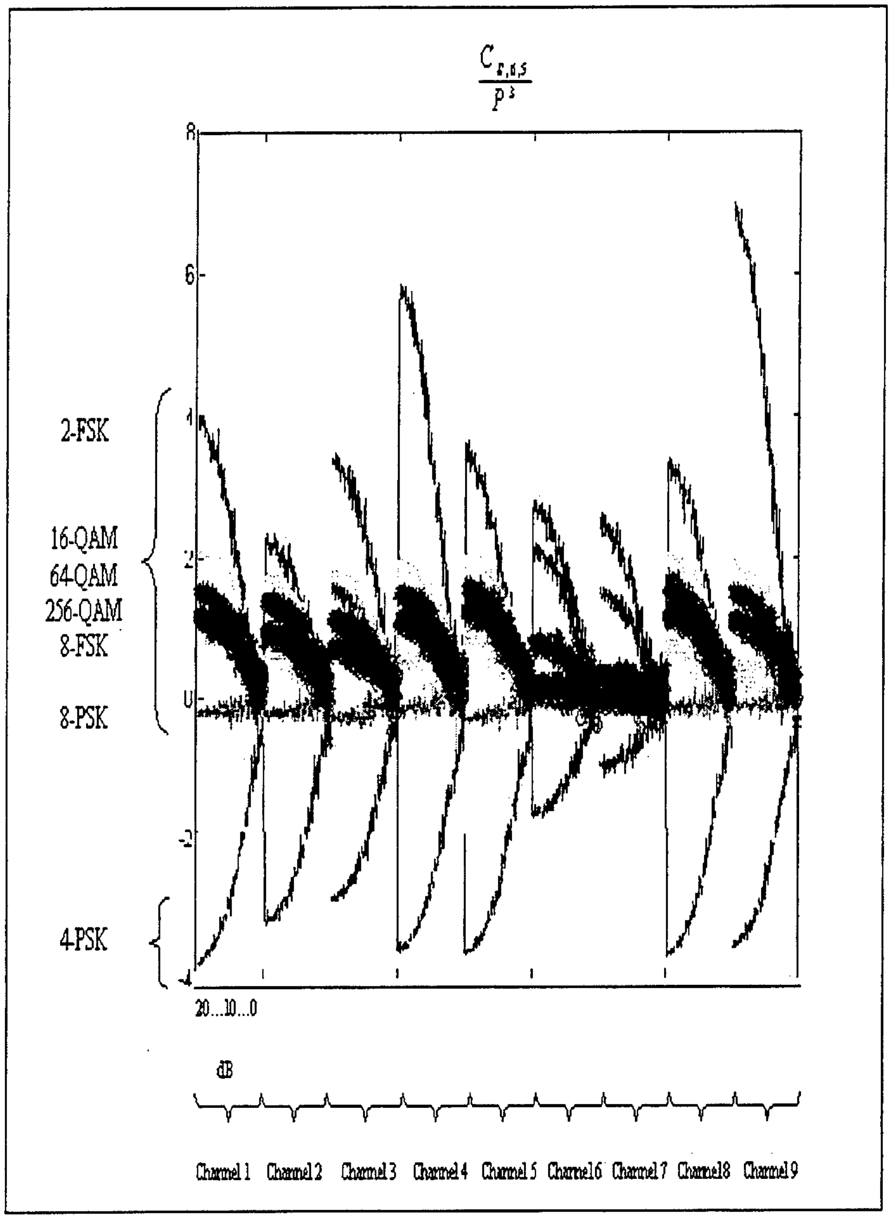


Figure E2-5. $\frac{C_{S,6,5}}{P^3}$, 1000 samples data-set, 100 trials per SNR level.

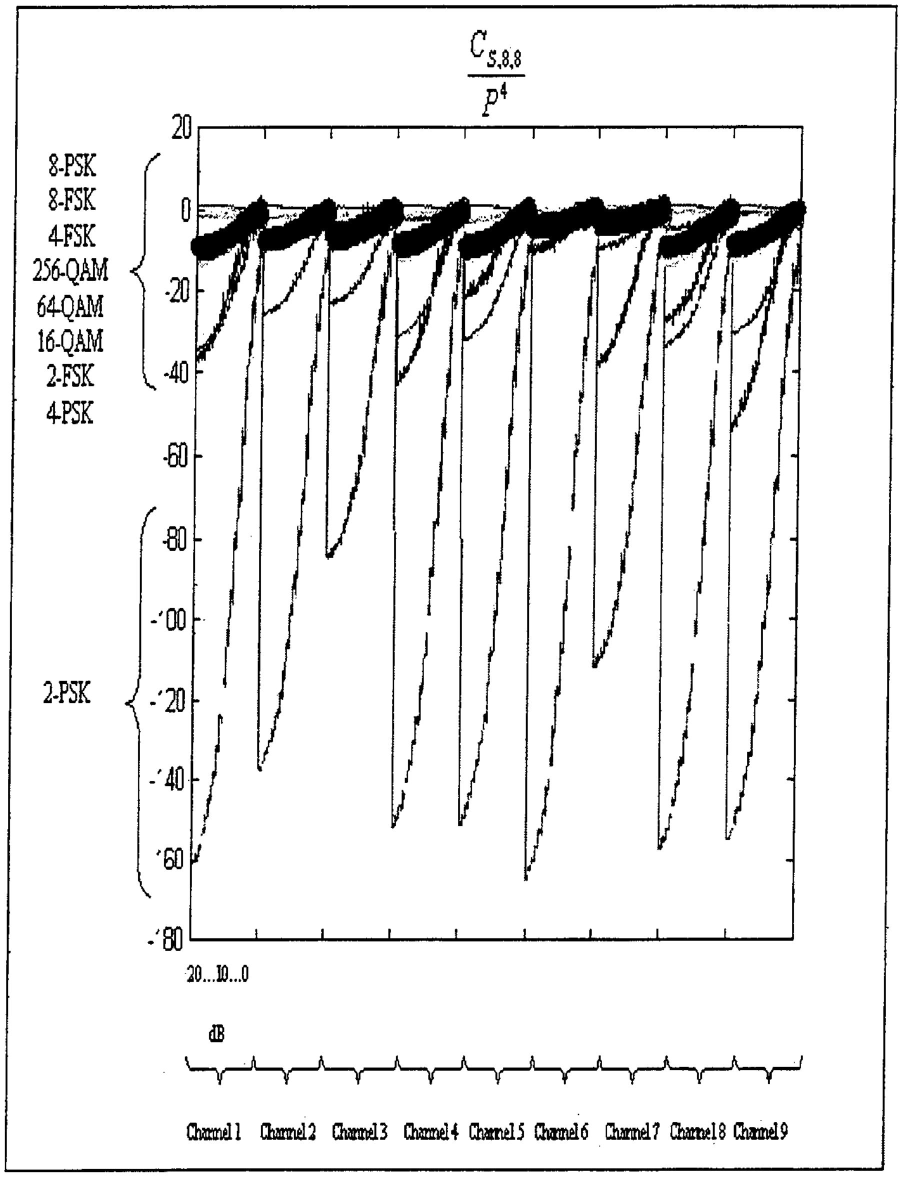


Figure E2-6. $\frac{C_{5,8,8}}{P^4}$, 15,000 samples data-set, 100 trials per SNR level.

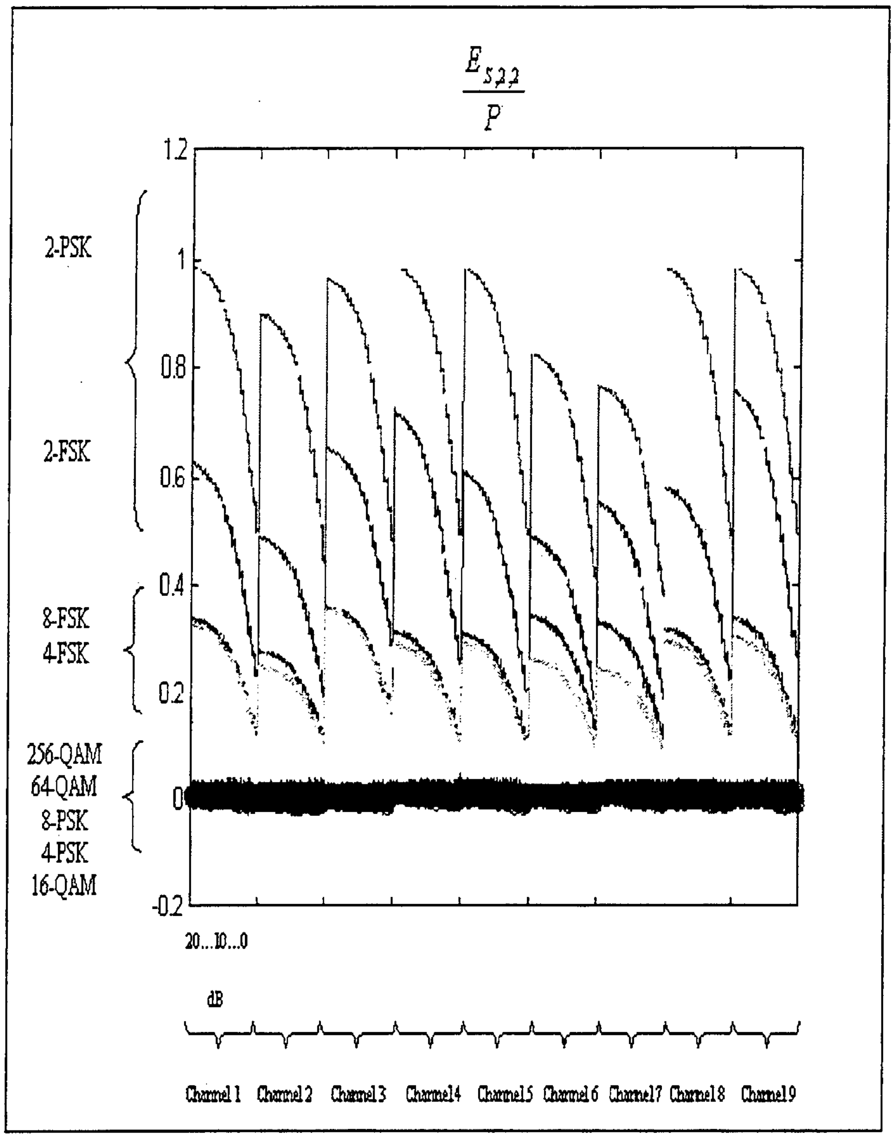


Figure E2-7. $\frac{E_{S,2,2}}{P}$, 15,000 samples data-set, 100 trials per SNR level.

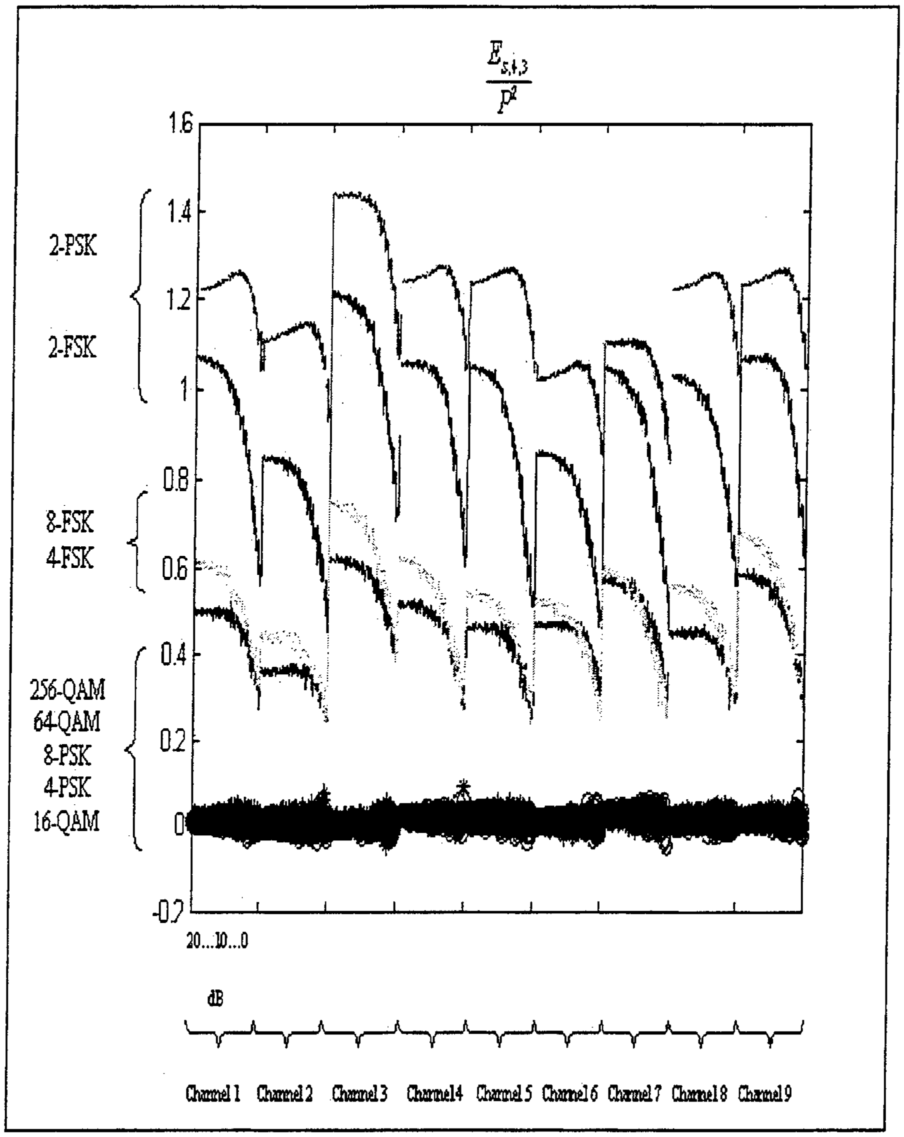


Figure E2-8. $\frac{E_{S,4,3}}{P^2}$, 15,000 samples data-set, 100 trials per SNR level.

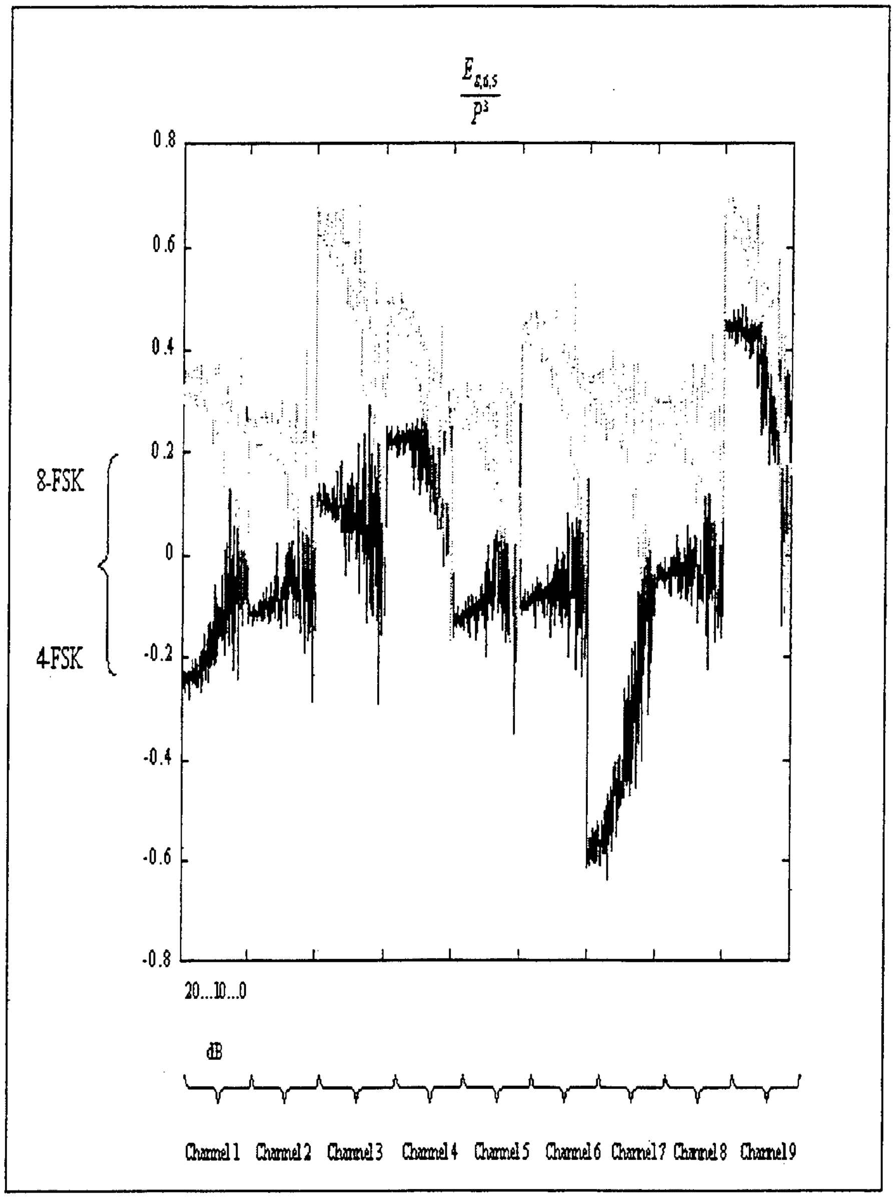


Figure E2-9. $\frac{E_{S,6,5}}{P^3}$, 15,000 samples data-set, 100 trials per SNR level.

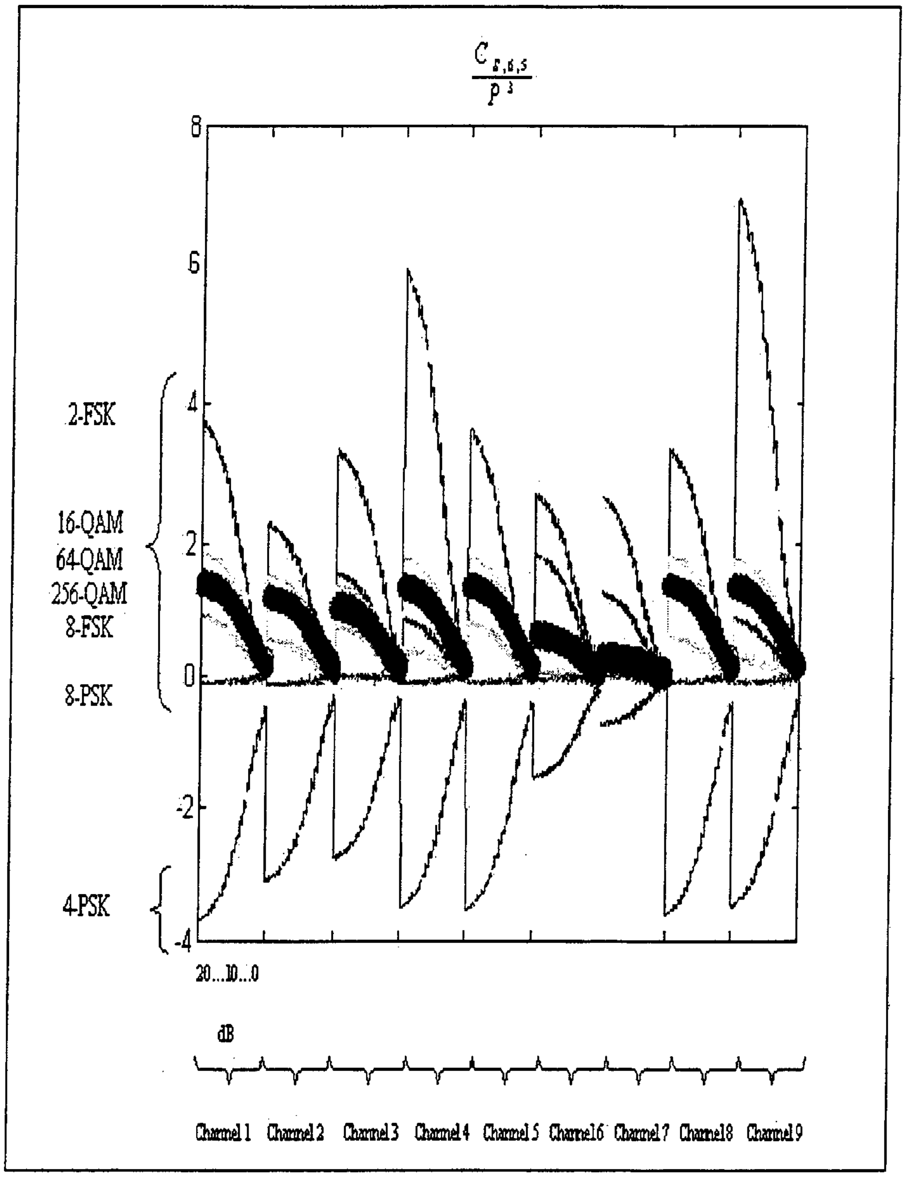


Figure E2-10. $\frac{C_{s,6,5}}{P^3}$, 15,000 samples data-set, 100 trials per SNR level.

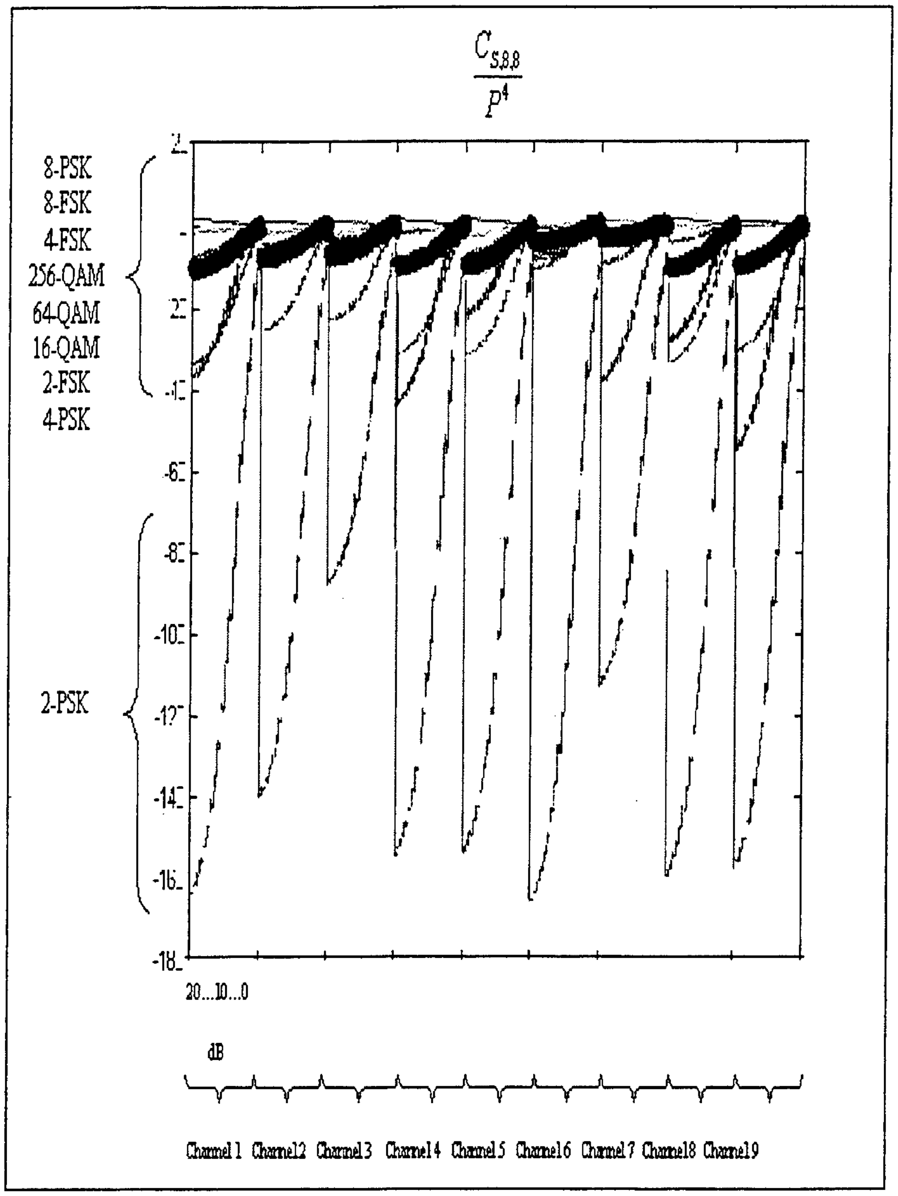


Figure E2-11. $\frac{C_{S,8,8}}{P^4}$, 30,000 samples data-set, 100 trials per SNR level.

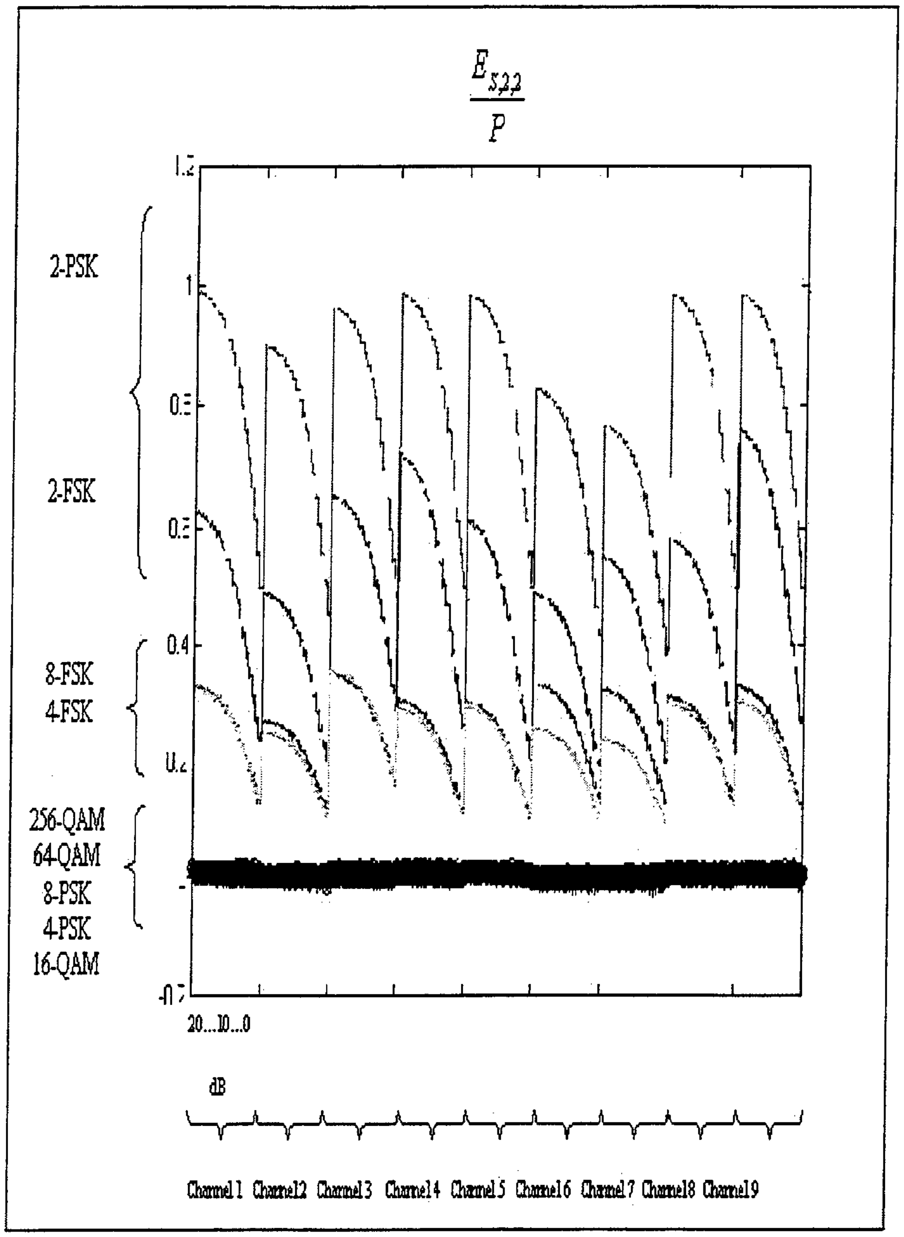


Figure E2-12. $\frac{E_{S,2,2}}{P}$, 30,000 samples data-set, 100 trials per SNR level.

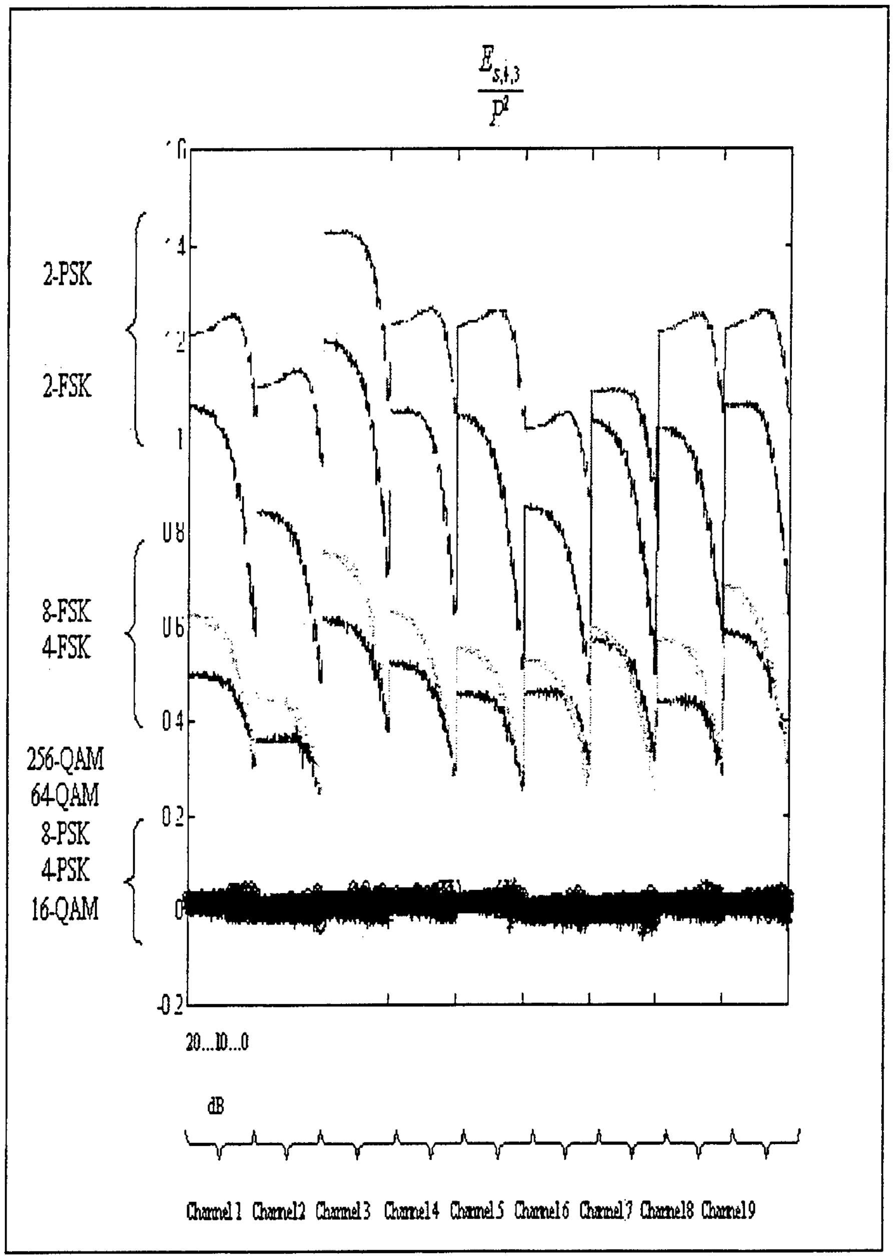


Figure E2-13. $\frac{E_{S,4,3}}{P^2}$, 30,000 samples data-set, 100 trials per SNR level.

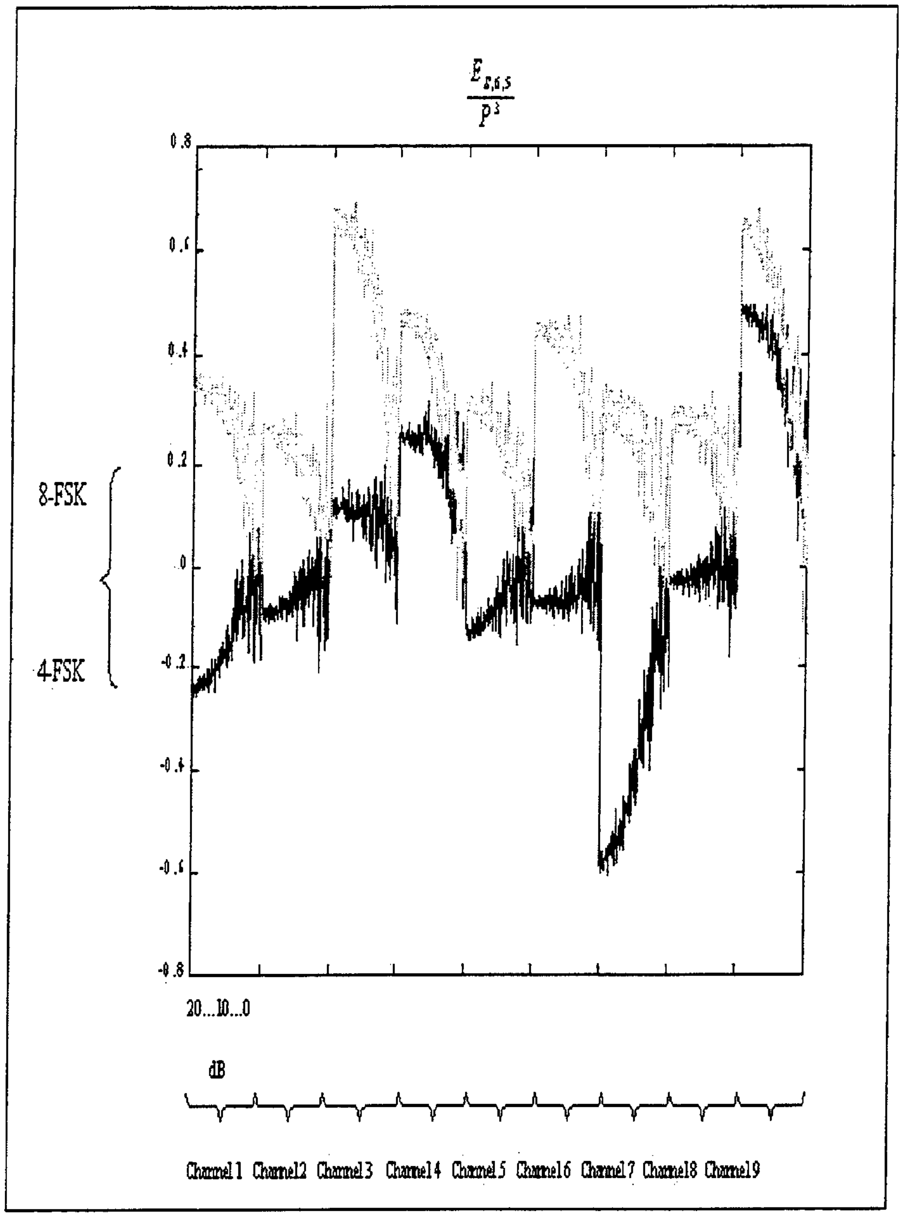


Figure E2-14. $\frac{E_{5,6,5}}{P^3}$, 30,000 samples data-set, 100 trials per SNR level.

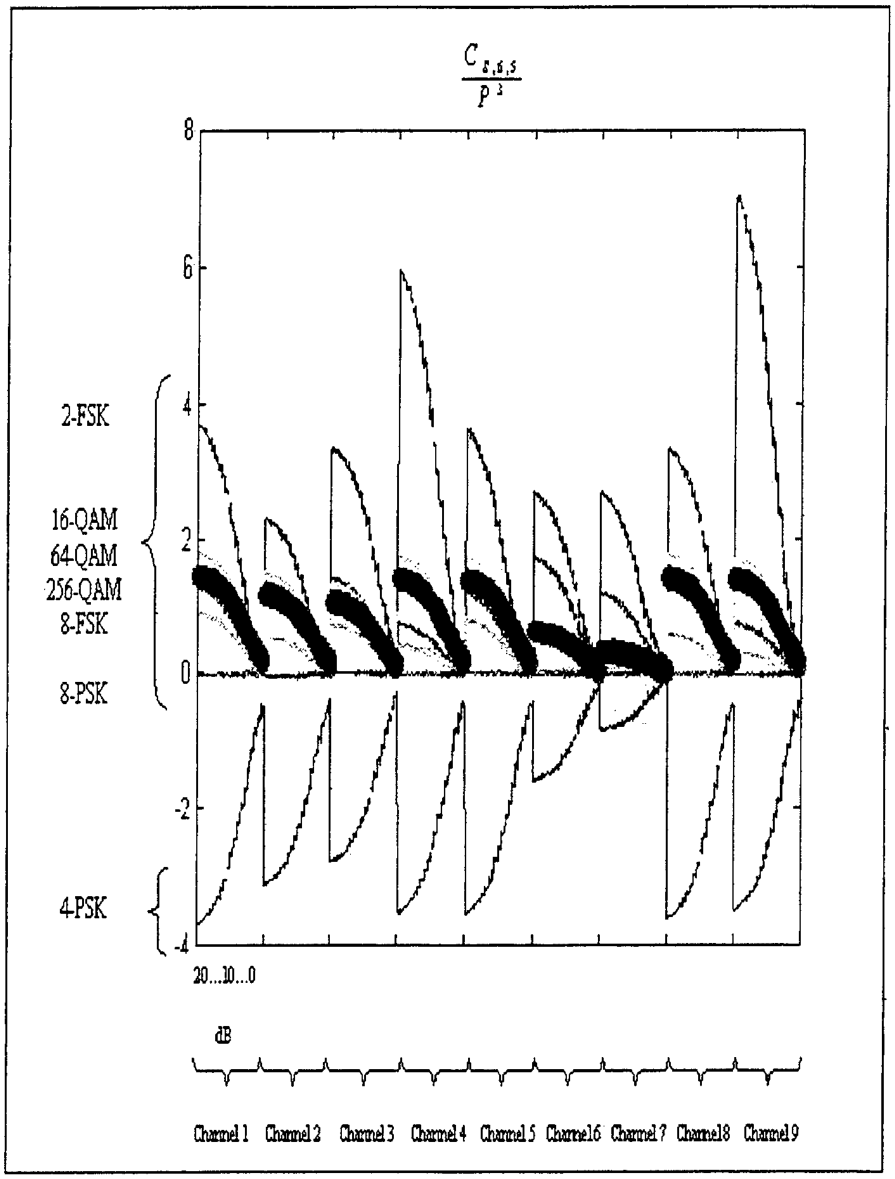


Figure E2-15. $\frac{C_{5,6,5}}{P^3}$, 30,000 samples data-set, 100 trials per SNR level.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F. SIMULATION RESULTS

Simulation results are divided into three main categories. The first category uses a rural area propagation model (Figure C-10). The second category a small town propagation model (Figure C-12) and the third category an urban propagation model with severe multi-path distortions (Figure C-15). Each category contains simulations of seven different signal- to-noise ratio levels from 20dB to 2dB. Fifty trials per SNR level and per category have been created, forming a total of twenty-one confusion matrixes. Figures C-22 to C-28 present the linear channel simulation case.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	50	0	0	0	0	0	0	0
8-FSK	0	0	50	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	46	3	1
64-QAM	0	0	0	0	0	0	0	50	0
256-QAM	0	0	0	0	0	0	0	0	50

Table F-1. Rural area propagation channel model, SNR=20dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	50	0	0	0	0	0	0	0
8-FSK	0	0	50	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	44	4	2
64-QAM	0	0	0	0	0	0	0	50	0
256-QAM	0	0	0	0	0	0	0	4	46

Table F-2. Rural area propagation channel model, SNR=17dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	50	0	0	0	0	0	0	0
8-FSK	0	0	50	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	22	19	9
64-QAM	0	0	0	0	0	0	0	50	0
256-QAM	0	0	0	0	0	0	0	1	49

Table F-3. Rural area propagation channel model, SNR=14dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	50	0	0	0	0	0	0	0
8-FSK	0	0	50	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	0	49	1
64-QAM	0	0	0	0	0	0	0	49	1
256-QAM	0	0	0	0	0	0	0	9	41

Table F-4. Rural area propagation channel model, SNR=11dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	50	0	0	0	0	0	0	0
8-FSK	0	0	50	0	0	0	0	0	0
2-PSK	40	0	0	10	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	0	31	19
64-QAM	0	0	0	0	0	0	0	35	15
256-QAM	0	0	0	0	0	0	0	30	20

Table F-5. Rural area propagation channel model, SNR=8dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	0	1	49	0	0	0	0	0	0
4-FSK	0	45	5	0	0	0	0	0	0
8-FSK	0	7	43	0	0	0	0	0	0
2-PSK	50	0	0	0	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	0	19	31
64-QAM	0	0	0	0	0	0	0	48	2
256-QAM	0	0	0	0	0	0	0	50	0

Table F-6. Rural area propagation channel model, SNR=5dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	0	2	48	0	0	0	0	0	0
4-FSK	0	41	9	0	0	0	0	0	0
8-FSK	0	30	20	0	0	0	0	0	0
2-PSK	2	0	0	0	0	48	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	0	50	0
64-QAM	0	0	0	0	0	0	0	50	0
256-QAM	0	0	0	0	0	0	0	50	0

Table F-7. Rural area propagation channel model, SNR=2dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	44	0	0	0	0	6	0	0	0
4-FSK	0	50	0	0	0	0	0	0	0
8-FSK	0	6	44	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	36	14	0
64-QAM	0	0	0	0	0	0	7	12	31
256-QAM	0	0	0	0	0	0	0	18	32

Table F-8. Small town propagation channel model, SNR=20dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	48	0	0	0	0	2	0	0	0
4-FSK	0	50	0	0	0	0	0	0	0
8-FSK	0	3	47	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	1	49	0	0	0
16-QAM	0	0	0	0	0	0	35	11	4
64-QAM	0	0	0	0	0	0	1	22	27
256-QAM	0	0	0	0	0	0	0	19	31

Table F-9. Small town propagation channel model, SNR=17dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	47	0	0	0	2	1	0	0	0
4-FSK	0	50	0	0	0	0	0	0	0
8-FSK	0	2	48	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	36	2	12
64-QAM	0	0	0	0	0	0	0	21	29
256-QAM	0	0	0	0	0	0	1	28	21

Table F-10. Small town propagation channel model, SNR=14dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	46	0	0	0	0	0	1	1	2
4-FSK	0	50	0	0	0	0	0	0	0
8-FSK	0	2	48	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	33	8	9
64-QAM	0	0	0	0	0	0	0	50	0
256-QAM	0	0	0	0	0	0	0	41	9

Table F-11. Small town propagation channel model, SNR=11dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	48	0	0	2	0	0	0	0	0
4-FSK	0	50	0	0	0	0	0	0	0
8-FSK	0	5	45	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	10	31	9
64-QAM	0	0	0	0	0	0	0	50	0
256-QAM	0	0	0	2	0	0	0	44	4

Table F-12. Small town propagation channel model, SNR=8dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	0	0	0	50	0	0	0	0	0
4-FSK	0	22	1	27	0	0	0	0	0
8-FSK	0	0	40	10	0	0	0	0	0
2-PSK	50	0	0	0	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	36	0	0	0	11	3
64-QAM	0	0	0	50	0	0	0	0	0
256-QAM	0	0	0	50	0	0	0	0	0

Table F-13. Small town propagation channel model, SNR=5dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	0	0	0	50	0	0	0	0	0
4-FSK	0	15	5	30	0	0	0	0	0
8-FSK	0	0	0	50	0	0	0	0	0
2-PSK	1	0	0	0	0	49	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	50	0	0	0	0	0
64-QAM	0	0	0	50	0	0	0	0	0
256-QAM	0	0	0	50	0	0	0	0	0

Table F-14. Small town propagation channel model, SNR=2dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	48	0	0	0	0	2	0	0	0
4-FSK	0	48	2	0	0	0	0	0	0
8-FSK	0	12	38	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	35	15	0
64-QAM	0	0	0	0	0	0	0	42	8
256-QAM	0	0	0	0	0	0	0	29	21

Table F-15. Urban area propagation channel model, SNR=20dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	40	0	0	0	0	10	0	0	0
4-FSK	0	48	2	0	0	0	0	0	0
8-FSK	0	14	36	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	24	26	0
64-QAM	0	0	0	0	0	0	0	44	6
256-QAM	0	0	0	0	0	0	0	37	13

Table F-16. Urban area propagation channel model, SNR=17dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	48	2	0	0	0	0	0	0
8-FSK	0	12	38	0	0	0	0	0	0
2-PSK	50	0	0	0	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	27	22	1
64-QAM	0	0	0	0	0	0	0	45	5
256-QAM	0	0	0	0	0	0	0	38	12

Table F-17. Urban area propagation channel model, SNR=14dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	45	5	0	0	0	0	0	0
8-FSK	0	19	31	0	0	0	0	0	0
2-PSK	50	0	0	0	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	35	15	0
64-QAM	0	0	0	0	0	0	0	49	1
256-QAM	0	0	0	0	0	0	0	49	1

Table F-18. Urban area propagation channel model, SNR=11dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	45	5	0	0	0	0	0	0
8-FSK	0	14	36	0	0	0	0	0	0
2-PSK	50	0	0	0	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	4	46	0
64-QAM	0	0	0	0	0	0	0	50	0
256-QAM	0	0	0	0	0	0	0	50	0

Table F-19. Urban area propagation channel model, SNR=8dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	0	1	49	0	0	0	0	0	0
4-FSK	0	37	13	0	0	0	0	0	0
8-FSK	0	16	34	0	0	0	0	0	0
2-PSK	50	0	0	0	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	0	50	0
64-QAM	0	0	0	0	0	0	0	50	0
256-QAM	0	0	0	0	0	0	0	50	0

Table F-20. Urban area propagation channel model, SNR=5dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	0	43	7	0	0	0	0	0	0
4-FSK	0	34	16	0	0	0	0	0	0
8-FSK	0	28	22	0	0	0	0	0	0
2-PSK	50	0	0	0	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	0	50	0
64-QAM	0	0	0	0	0	0	0	50	0
256-QAM	0	0	0	0	2	0	0	48	0

Table F-21. Urban area propagation channel model, SNR=2dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	49	1	0	0	0	0	0	0
8-FSK	5	15	30	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	50	0	0
64-QAM	0	0	0	0	0	0	1	39	10
256-QAM	0	0	0	0	0	0	3	9	38

Table F-22. Linear channel model c=[1,0,0.5], SNR=20dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	50	0	0	0	0	0	0	0
8-FSK	1	18	31	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	50	0	0
64-QAM	0	0	0	0	0	0	1	34	15
256-QAM	0	0	0	0	0	0	0	6	44

Table F-23. Linear channel model c=[1,0,0.5], SNR=17dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	49	1	0	0	0	0	0	0
8-FSK	0	25	25	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	46	0	4
64-QAM	0	0	0	0	0	0	1	10	39
256-QAM	0	0	0	0	0	0	0	8	42

Table F-24. Linear channel model c=[1,0,0.5], SNR=14dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	39	11	0	0	0	0	0	0
8-FSK	0	18	32	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	2	0	0	48	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	24	3	23
64-QAM	0	0	0	0	0	0	0	13	37
256-QAM	0	0	0	0	0	0	0	45	5

Table F-25. Linear channel model c=[1,0,0.5], SNR=11dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	25	25	0	0	0	0	0	0
8-FSK	0	26	24	0	0	0	0	0	0
2-PSK	0	0	0	50	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	0	4	46
64-QAM	0	0	0	0	0	0	0	50	0
256-QAM	0	0	0	0	0	0	0	50	0

Table F-26. Linear channel model c=[1,0,0.5], SNR=8dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	50	0	0	0	0	0	0	0	0
4-FSK	0	37	13	0	0	0	0	0	0
8-FSK	0	17	33	0	0	0	0	0	0
2-PSK	50	0	0	0	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	0	0	0	50	0
64-QAM	0	0	0	0	0	0	0	50	0
256-QAM	0	0	0	0	0	0	0	50	0

Table F-27. Linear channel model c=[1,0,0.5], SNR=5dB, 50 trials.

	2-FSK	4-FSK	8-FSK	2-PSK	4-PSK	8-PSK	16-QAM	64-QAM	256-QAM
2-FSK	0	50	0	0	0	0	0	0	0
4-FSK	0	42	8	0	0	0	0	0	0
8-FSK	0	23	27	0	0	0	0	0	0
2-PSK	50	0	0	0	0	0	0	0	0
4-PSK	0	0	0	0	50	0	0	0	0
8-PSK	0	0	0	0	0	50	0	0	0
16-QAM	0	0	0	0	34	0	0	16	0
64-QAM	0	0	0	0	22	26	0	2	0
256-QAM	0	0	0	0	26	24	0	0	0

Table F-28. Linear channel model c=[1,0,0.5], SNR=2dB, 50 trials.

LIST OF REFERENCES

[ANA95] E. Azzouz and A. Nandi, "Automatic Identification of Digital Modulation Types," University of Strachclyde, Glasgow, UK, 1995

[BAR00] S. Barbarossa, A. Swami, B. Sadler and G. Spadafora, "Classification of Digital Constellations under Unknown Multi-path Propagation Conditions," *Proceedings of SPIE*, Vol. 4045, pp. 175-185, 2000

[BER77] R. Beran, "Minimum Hellinger Distance Estimates for Parametric Models," *Annals of Statistics*, Vol. 5, pp. 445-463, 1977

[BSC98] S. Barbarossa and A. Scaglione, "Blind Equalization using Cost Function Matched to the Signal Constellation," *Proceedings of the 31st Asilomar Conference*, Vol. 1, pp. 550-554, 1998

[CJJ00] W. Chung and C. Johnson Jr, "Characterization of the Regions of Convergence of CMA Adapted Blind Fractionally Spaced Equalizer," 1998, Applied Signal Technology, Inc, www.appsig.com/ papers/1828d/828d_1.html, accessed on 12/13/2000

[COP00] Co-Optic Inc. Specifications for the COm4002 Rack Mounted Demodulator, http://www.co-optic.com/com4002.htm, accessed on 1/10/2001

[EVA00] B. Evans, http://www.ece.utexas.edu/~bevans/ courses/realtime/ lectures/16_Matched_Filtering/lecture16/tsld015.htm, University of Texas at Austin, accessed on 13/11/2000

[GRE00] M.Greenman, "Fuzzy Modes and Modern Digital Modes," http://www.qsl.net/zl1bpu/FUZZY/MFSK.html, accessed on 11/08/2000

[HAA96] R. Haas, "Applications of multicarrier modulation in mobile radio communications", PhD thesis, Ecole Nationale Superieure des Telecommunications, Paris, 1996, http://www.baltzer.nl/wicom/demo99/ chaptr03/channel.asp, accessed on 2/12/2000

[HAD98] X. Huo and D. Donoho, "A Simple and Robust Modulation Classification Method via Counting," *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 6, pp. 3289-3292, 1998

[HAH99] L. Hong and K. Ho, "Identification of Digital Modulation Types using the Wavelet Transform," *Proceedings of the 1999 IEEE Military Communications Conference*, Vol. 1, pp. 427-431, 1999

[HAJ99] M. Haun and D. Joned, "The Fractionally Spaced Vector Constant Modulus Algorithm," *Proceedings of the 1999 International Conference on Acoustic, Speech and Signal Processing*,Vol. 5 , pp. 2647-2650, 1999

[HAY96] S. Haykin, *Adaptive filter theory*, Prentice Hall, 3rd edition, 1996

[HDB96] M. Hagan, H. Demuth, M. Beale, *Neural network design*, PWS, 1st edition, 1996

[HPC95] K. Ho, W. Prokopiw and Y. Chan, "Modulation Identification by the Wavelet Transform," *Proceedings of the 1995 IEEE Military Communications Conference*, Vol. 2, pp. 886-890, 1995

[HYY00] A. Hyvärinen , http://www.cis.hut.fi /~aapo/papers/ NCS99web/ node51.html, Laboratory of Computer and Information Science, University of Technology, Helsinki, accessed on 12/23/2000

[JAO98] R. Johnson, P. Schniter, T. Endres, J. Behm, D. Brown and R. Casas, "Blind Equalization using the Constant Modulus criterion: A review, " *IEEE Proceedings*, Vol. 86, No. 10, pp. 1927-1950, 1998

[KJC99] H. Ketterer, F. Jondral and A. Costa, "Classification of Modulation Modes using Time-Frequency Methods," *IEEE Proceedings of the 1999 International Conference*, Vol. 5, pp.2471-2474, 1999

. [LAU94] D. Laurenson, "Indoor Radio Channel Propagation Modelling by Ray Tracing Techniques", PhD thesis, University of Edinburgh, United Kingdom, 1994, http://www.ee.ed.ac.uk/~dil/thesis_mosaic/section2_7_1.html, accessed on 3/22/2001

[MAB97] C. Martret and D. Boiteau, "Modulation Classification by means of different Orders Statistical Moments," *Proceedings of the 1997 IEEE Military Communications Conference*, Vol. 3, pp. 1387-1391, 1997

[MAR98] P. Marchand, *Détection et reconnaissance de modulations numériques à l'aide des statistiques cycliques d'ordre supérieur*, Ph. D. Dissertation, Institut National Polytechnique de Grenoble, 1998

[MML97] P. Marchand, C. Martret and J. Lacoume, "Classification of Linear Modulations by a Combination of Different Orders Cyclic Cumulants," *Proceedings of the IEEE Signal Processing Workshop*, pp. 47-51, 1997

[MOB00] B. Mobasseri, "Digital Modulation Classification using Constellation Shape," *Signal Processing*, Vol. 80, 2000, pp. 251-277

[MPR00] Virginia Tech's Mobile & Portable Radio Research Group (MPRG), http://www.mprg.ee.vt.edu/research/glomo/node7.htm, accessed on 11/18/2000

[PLC96] K. Chugg, C. Long and A. Polydoros, "Combined Likelihood Power Estimation and Multiple Hypothesis Modulation Classification," *Proceedings of the 29th Asilomar Conference on Signals, Systems, and Computers*, Vol. 2, pp. 1137-1141, 1996

[PRO95] J.Proakis, *Digital communications,* 3rd edition, McGraw Hill, 1995

[RAP99] T.Rappaport, *Wireless communications*, Prentice Hall, 2nd edition, 1999

[RIC00] Signal Processing Information Base, Rice University, Houston TX, http://spib.ece.rice.edu/spib/data/signals, accessed on 11/12/00

[ROS95] T.Ross, *Fuzzy logic with engineering applications*, McGraw-Hill, 1995

[SAH92] S. Soliman and S. Hsue, "Signal Classification using Statistical Moments," *IEEE Transactions on Communications*, Vol. 40, No. 5, pp. 908-916, 1992

[SAS00] A. Swami and B. Sadler, "Hierarchical Digital Modulation Classification using Cumulants," *IEEE Transactions on Communications*, Vol. 48 No. 3, pp. 416-428, 2000

[SCH00] P. Schniter, Blind Equalization Research Group, Cornell University http://www.backhoe.ee.cornell.edu/BERG/downloads/tutorials/CMA_FSE_surf/CMA_FSE_surf.html, accessed on 12/15/2000

[WIL99] J.Williams, Ashton University 1999, http://www.eeap.aston.ac.uk/teltec/tutorials/Digital%20Passband%20Transmission/notes/Coherent%20Binary%20FSK.htm, accessed on 11/23/2000

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center..2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, Ca 93943-5121

3. Chairman, Code EC 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, Ca 93943-5121

4. Engineering and Technology Curricular Office, Code 34...........................1
   Naval Postgraduate School
   Monterey, Ca 93943-5109

5. Prof. Monique P. Fargues, Code EC/Fa 3
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, Ca 93943-5121

6. Prof. T. Ha, Code EC/Ha 2
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, Ca 93943-5121

7. Prof. R. Cristi, Code EC/Cx 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, Ca 93943-5121

8. Hellenic Navy General Staff...2
   Department B/2, Stratopedo Papagoy, Mesogeion 151,
   Holargos, 155-00, Athens,
   Greece

9. Lt. Jg George Hatzichristos...1
   Tzavella 1, Palaio Faliro, 175-63, Athens,
   Greece